Universidad Carlos III de Madrid

TRABAJO FIN DE GRADO

PROGRAMACIÓN DE BRAZO ROBÓTICO PARA DEMOSTRACIÓN DE TERAPIA DE REHABILITACIÓN MOTORA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

UNIVERSIDAD CARLOS III DE MADRID

Escuela Politécnica Superior

Autor: Dennys Francisco Machasilla Sánchez

Tutor: Edwin Daniel Oña Simbaña

Septiembre 2016

AGRADECIMIENTOS

En primer lugar agradezco y dedico este trabajo fin de grado a mis padres y mi hermano, por la comprensión, motivación y apoyo que me han brindado para conseguir todas mis metas y en especial durante el desarrollo de este trabajo. Sin duda no habría llegado hasta aquí sin ellos.

También quiero agradecer especialmente a mi tutor Edwin la oportunidad de realizar este proyecto y aprender de él, y tanto al departamento de Ingeniería de Sistemas y Automática como al grupo de investigación Robotics Lab el haberme permitido realizarlo.

Gracias a todos.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

RESUMEN

Este trabajo forma parte de un proyecto desarrollado por la Universidad Carlos III de Madrid, que se llama RoboHealth-A, y que consiste en el desarrollo de robots de asistencia y rehabilitación para la mejora del bienestar de los pacientes en un ambiente hospitalario. El objetivo principal de dicho proyecto es desarrollar ayudas técnicas de rehabilitación y robots inteligentes en espacios hospitalarios, contribuyendo de esta manera, a la mejora del sistema de sanidad nacional. Los pacientes a los que está dirigido el proyecto se componen de personas con movilidad limitada, capacidades cognitivas reducidas y enfermedades crónicas; que representan aproximadamente el 10% de la población actual de la Unión Europea, teniendo un alto grado de dependencia de sus familiares. La aplicación de las tecnologías robóticas puede ayudar a estas personas a obtener una mejor calidad de vida, dándoles una mayor independencia.

La principal herramienta que se utilizará, o sobre la que se va a enfocar este trabajo fin de grado es ASIBOT, que es manipulador robótico con 5 grados de libertad, unos 10 kg de peso, de 1,3 m de alcance y 2kg de carga útil. ASIBOT ha sido desarrollado en la Universidad Carlos III de Madrid y presenta una serie de ventajas, inusuales en otros robots asistenciales. Debido a su ligero peso y carácter escalador, ASIBOT posee la capacidad de trasladarse entre diferentes puntos de anclaje situados en el entorno o montado sobre una silla de ruedas y realizar tareas en zonas de difícil acceso desde una base móvil. Véase por ejemplo [22]. Estas estaciones de anclaje permiten no sólo el desplazamiento y la acometida de tareas a través de superficies normalmente no utilizadas, como pueden ser suelos y paredes, sino el suministro de la alimentación necesaria al robot. Cuando el robot se encuentra acoplado a la estación de anclaje de la silla de ruedas, ver Figura 1, las propias baterías de la silla de ruedas se encargan de suministrar la energía para su funcionamiento.

Como bien dice el título de este trabajo, se va a desarrollar una tarea de programación de brazo robótico para ejecutar una demostración de rehabilitación motora. La programación de las trayectorias que debe seguir el brazo robótico se realizarán a través de una interfaz web construida específicamente para ASIBOT y de esta manera llevar a cabo una interacción humano-robot más sencilla y natural para los usuarios.

ABSTRACT

This work is part of a project developed by the University Carlos III of Madrid, called RoboHealth-A. It consists in the development of assistive and rehabilitation robots to improve the welfare of patients in smart hospital spaces. The main objective of this project is to develop rehabilitation aids and intelligent robots in hospital spaces, thus contributing to the improvement of the national health system. Patients who directed the project is made up of people with limited mobility, reduced cognitive abilities and chronic illnesses; representing approximately 10% of the current population of the European Union, having a high degree of dependence on their families. The application of robotic technologies can help these people get a better quality of life, giving them more independence.

The main tool to be used, or which will focus this end-of-degree work is ASIBOT, which is robotic manipulator with 5 degrees of freedom, about 10 kg, 1.3 m range and 2kg useful load. ASIBOT has been developed at the University Carlos III of Madrid and has a number of advantages, unusual in other assistive robots. Because of its light weight and character climber, ASIBOT has the ability to move between different anchor points in the environment or mounted on a wheelchair and perform tasks in areas of difficult access from a mobile base. See for example [22]. These docking stations not only allow displacement and rush through tasks normally unused surfaces, such as floors and walls, but supply the necessary power to the robot. When the robot is coupled to the docking station of the wheelchair, see Figure 1, the batteries in the wheelchair are responsible for providing energy for operation.

As stated by the title of this work, it will develop a programming task robotic arm to perform a demonstration of motor rehabilitation. Programming paths to follow the robotic arm will be done through a web interface built specifically for ASIBOT and thus carry out a easier and natural human-robot interaction for users.

Figura 1. Manipulador robótico ASIBOT

1 INTRODUCCIÓN

El objetivo principal de este capítulo es dar al lector una visión general sobre el trabajo desarrollado, explicando su contexto, los objetivos marcados y por último se incluye una sección donde se define la estructura del documento.

1.1 Contexto

Este trabajo se encuentra inmerso dentro del área de la robótica asistencial, que consiste en prestar ayuda y apoyo a un usuario humano con necesidades especiales a través de una interacción física, o no necesariamente.

La robótica asistencial se está desarrollando como apoyo a las personas mayores y discapacitadas en actividades de la vida cotidiana realizadas en entornos no estructurados del mundo real. Una de las características principales en la interacción humano-robot (HRI, Human-Robot Interaction), en robótica asistencial, es la diversidad de los potenciales usuarios, con diferentes tipos y grados de discapacidad, tanto física como mental. Los manipuladores robóticos asistenciales presentan la capacidad de mejorar la calidad de vida de los usuarios finales mediante el aumento del nivel de independencia en la ejecución de dichas actividades, que abarcan desde el posicionamiento y emplazamiento de objetos hasta tareas que involucran el cuidado personal [21].

Precisamente el campo de la Interacción Humano-Robot (HRI) ha aumentado en gran medida durante los últimos años, debido a la complejidad de los sistemas y plataformas robóticas para el control. Esto es más notable en los sistemas desarrollados para las personas con discapacidad, como los sistemas robóticos de asistencia, donde la HRI debe ser diseñada teniendo en cuenta el tipo de usuarios que vayan a utilizar el sistema, así como sus discapacidades. En los últimos años, los avances innovadores se han desarrollado en este campo gracias al desarrollo de interfaces multimodales que pueden adaptarse a las necesidades de los usuarios de estos sistemas.

Cabe destacar que la robótica asistencial fuera del entorno cerrado de la industria y dirigida a interactuar con personas, por ejemplo, con ancianos o niños enfermos en hospitales para entretenerlos está en pleno auge y España es puntera con varios grupos de investigación y desarrollo del más alto nivel en esa área [19], como por ejemplo “Robotics Lab” de la Universidad Carlos III de Madrid. Y es por eso precisamente que he elegido formar parte de este proyecto, porque a pesar de que los avances en robótica de servicios o asistencial son

muy sorprendentes, se reconoce que todavía queda mucho por hacer y descubrir, ya que este tipo de tecnología está en sus comienzos.

Las tecnologías robóticas asistenciales tradicionales se han centrado, principalmente, en el desarrollo de tres conceptos:

- Sistemas estáticos que operan en ambientes estructurados.
- Sistemas robóticos montados sobre silla de ruedas.
- Manipuladores móviles que acompañan al usuario y desempeñan aplicaciones de cuidado personal.

El primer tipo de sistema robótico es recomendable cuando el usuario necesita asistencia para desarrollar siempre las mismas aplicaciones, dentro de un mismo entorno doméstico de espacio reducido. Estas tareas se corresponden, a modo de ejemplo, con las relacionadas con la higiene personal, comer, beber, etc. El manipulador robótico Handy 1 (Topping, 1993), mostrado en la Figura 2, es un ejemplo de sistema robótico estático de bajo coste para la asistencia y el cuidado personal. No obstante, los sistemas robóticos estáticos presentan el gran inconveniente que implica el cambio de emplazamiento de la plataforma robótica. En ocasiones, este cambio puede resultar muy complicado o imposible.

Figura 2. Manipulador robótico Handy 1

Otro tipo de robots utilizados en rehabilitación son los sistemas robóticos montados sobre silla de ruedas. En lo que concierne a esta modalidad de robots, el sistema robótico MANUS es uno de los más importantes e influyentes. Este sistema consta de un brazo robótico, dotado de seis grados de libertad, que permanece permanente anclado en el lado derecho o izquierdo de una silla de ruedas eléctrica. Sin embargo, esta localización asimétrica del manipulador puede convertirse en un inconveniente durante la ejecución de ciertas tareas y, asimismo, producir problemas de movilidad cuando se atraviesan puertas o existen escaleras. En la Figura 3 se puede observar el sistema robótico MANUS.

Figura 3. Brazo robótico MANUS montado sobre silla de ruedas.

El tercer concepto se refiere a un sistema móvil con manipulador robótico integrado que permite realizar el seguimiento de la silla de ruedas utilizada por el usuario. Este concepto tiene inconvenientes similares a los ya expuestos con anterioridad. La movilidad dentro de un entorno doméstico no es siempre ideal debido a la existencia de escalones u otros obstáculos. Sin embargo, introduce una nueva ventaja que no presentan los modelos anteriores, el robot tiene la capacidad de desplazarse de forma independiente con respecto a la silla de ruedas y al usuario. Un ejemplo bastante conocido de este tipo de sistema robótico es el manipulador móvil KARES II (Bien, Z. et al., 2004), mostrado en la Figura 4.

Figura 4. Integración del robot KARRES mediante el uso de dos plataformas: sistema móvil con robot KARES II y silla de ruedas eléctrica.

Es verdad que todavía existen limitaciones en cuanto a la interacción de los robots con el mundo físico dado que no perciben el mundo como los humanos. Algunas de esas limitaciones, sobre las cuales se están realizando muchas investigaciones y avances son:

- **La manipulación compleja de objetos**. En el caso del hombre la mano se mueve gracias al cerebro, pero dotar a los robots de ese órgano que mueva sus extremidades y también mantenga el equilibrio es muy complicado.

- **Eficiencia energética**. La autonomía energética es muy importante en el área de la robótica, ya que se necesita de mucha energía para mover un robot con tantos motores.

- **Locomoción**. En robótica la solución más usada son las ruedas, pero eso limita a un desplazamiento en suelo plano. Por tanto, se vuelve a necesitar de un órgano (cerebro) que sea capaz de mover extremidades y mantener el equilibrio.

- **Percepción**. Para poder interactuar con el medio físico es necesario poder percibirlo e identificarlo cosa que para los humanos es fácil, pero no así para los robots.

1.2 Objetivos

El principal objetivo de este trabajo es programar un brazo robótico para que sea capaz de realizar una demostración de terapia de rehabilitación motora. Para ello se establecen otros subobjetivos, los cuales se detallan a continuación, que permitan conseguir este fin último.

Cabe destacar, que la mayoría de recursos utilizados para dicha programación ya están creados pero están desactualizados, entonces el objetivo secundario es comprender el trabajo hecho anteriormente y habilitarlo para su uso actual.

A continuación se desglosan los subobjetivos marcados:

- Realización de un estudio del estado del arte en metodologías de programación de robots asistenciales.

- Habilitación o puesta en marcha de la interfaz web de ASIBOT; a través de la cual se va a realizar la comunicación con el simulador o incluso con el robot real, para la programación del mismo.

- Demostración del manejo del simulador, mostrando la comunicación entre la interfaz web y el simulador.

- Elaboración de trayectorias para simular ejercicios utilizados en terapia de rehabilitación de miembro superior, para valorar su posterior ejecución con el robot real.

La utilización de la interfaz web, ofrece una manera sencilla y eficaz de programar trayectorias para usuarios no expertos en robótica. En este caso, dentro del proyecto

RoboHealth, los usuarios de la interfaz web serán los terapeutas, quienes utilizando la interfaz podrán diseñar y modificar trayectorias que se ejecutarán con el robot real en ejercicios de rehabilitación motora.

1.3 Estructura del documento

El presente documento está dividido en diversos capítulos que reúnen toda la información concerniente al desarrollo de este trabajo. A continuación se procede a resumir el contenido de cada uno de ellos.

- En el capítulo 1: “Introducción” se ha realizado una presentación del trabajo a desarrollar, exponiendo el contexto actual de la robótica de servicios o asistencial y los objetivos marcados.
- El capítulo 2: “Estado del arte” contiene una visión global de las tecnologías y herramientas estrechamente relacionadas con este trabajo.
- En el capítulo 3: “Desarrollo del trabajo” se mencionan y explican las herramientas necesarias para la consecución de los objetivos.
- En el capítulo 4: “Resultados” se realiza un balance del rendimiento que es capaz de brindar el trabajo realizado y se recogen los resultados de su implementación.
- En el capítulo 5: “Conclusiones y trabajos futuros” se recogen las conclusiones extraídas a lo largo del desarrollo del trabajo. Además se describen los conocimientos aprendidos y las líneas futuras de ampliación y mejora.
- Por último, se enumeran las diferentes referencias a libros, artículos y webs utilizadas.

2 ESTADO DEL ARTE

El principal objetivo de este capítulo es realizar un pequeño análisis sobre los conceptos, tecnologías y herramientas que guardan cierta relación con el presente trabajo. Se busca poder facilitar al lector la comprensión de ciertos conceptos y del resto de los capítulos.

2.1 Origen y situación actual de la robótica

La palabra robot surge de una obra de teatro que se llama “Robots Universales de Rossum” (RUR) y cuyo autor es Karel Capek. El significado de esta palabra en checoslovaco es “trabajador o sirviente". Por tanto, se puede afirmar que el término “robot” proviene de la ciencia ficción. Sin embargo se puede encontrar en multitud de mitos de diferentes culturas una referencia a la posibilidad de crear un ente con inteligencia, desde el Popol-Vuh del pueblo maya hasta el Golem del judaísmo.

La robótica es una ciencia o rama de la tecnología, que estudia el diseño y construcción de máquinas que tienen un propósito general, y capacidades similares o superiores que el ser humano o que requieren del uso de la inteligencia.

Un sistema Robótico puede describirse, como "Aquel que es capaz de recibir información, de comprender su entorno a través del empleo de modelos, de formular y de ejecutar planes, y de controlar o supervisar su operación". La Robótica es esencialmente multidisciplinar y se apoya en gran medida en los progresos de la microelectrónica y de la informática, así como en los de nuevas disciplinas tales como el reconocimiento de patrones y de inteligencia artificial.

Un Robot es un dispositivo generalmente mecánico, que desempeña tareas automáticamente, ya sea de acuerdo a supervisión humana directa, a través de un programa predefinido o siguiendo un conjunto de reglas generales, utilizando técnicas de inteligencia artificial. Generalmente estas tareas reemplazan, asemejan o extienden el trabajo humano, como ensamble en líneas de manufactura, manipulación de objetos pesados o peligrosos, trabajo en el espacio, etc.

En resumen se puede decir que las características principales de un robot son:

- Manejar objetos, o sea un manipulador. Un robot se diseña con este fin, teniendo en cuenta que ha de ser muy versátil a la hora de utilizar herramientas y manejarlas.
- La diferencia de otras máquinas automáticas es su capacidad para realizar trabajos completamente diferentes adaptándose al medio, e incluso pudiendo

tomar decisiones. A eso es a lo que se refiere lo de multifuncional y reprogramable.

No hay una clasificación estandarizada de robots puesto que es un tema muy peliagudo ya que no hay un acuerdo común en cuántos y cuáles son los tipos de robots y sus características esenciales. Pero la más común es la que se presenta a continuación [20]:

Según su cronología:

- **1ª Generación**. Manipuladores. Son sistemas mecánicos multifuncionales con un sencillo sistema de control, bien manual, de secuencia fija o de secuencia variable.

- **2ª Generación**. Robots de aprendizaje. Repiten una secuencia de movimientos de movimientos que ha sido ejecutada previamente por un operador humano. El modo de hacerlo es a través de un dispositivo mecánico. El operador realiza los movimientos requeridos mientras el robot le sigue y los memoriza.

- **3ª Generación**. Robots con control sensorizado. El controlador es una computadora que ejecuta las órdenes de un programa y las envía al manipulador para que realice los movimientos necesarios.

- **4ª Generación**. Robots inteligentes. Son similares a los anteriores, pero además poseen sensores que envían información a la computadora de control sobre el estado del proceso. Esto permite una toma inteligente de decisiones y el control del proceso en tiempo real.

Según su estructura:

La estructura, es definida por el tipo de configuración general del Robot, puede ser metamórfica. El concepto de metamorfismo, de reciente aparición, se ha introducido para incrementar la flexibilidad funcional de un Robot a través del cambio de su configuración por el propio Robot. El metamorfismo admite diversos niveles, desde los más elementales (cambio de herramienta o de efecto terminal), hasta los más complejos como el cambio o alteración de algunos de sus elementos o subsistemas estructurales. Los dispositivos y mecanismos que pueden agruparse bajo la denominación genérica del Robot, tal como se ha indicado, son muy diversos y es por tanto difícil establecer una clasificación coherente de los mismos que resista un análisis crítico y riguroso. La subdivisión de los Robots, con base en su arquitectura, se hace en los siguientes grupos: poliarticulados, móviles, androides, zoomórficos e híbridos.

- **Poliarticulados**. En este grupo se encuentran los Robots de muy diversa forma y configuración, cuya característica común es la de ser básicamente sedentarios (aunque excepcionalmente pueden ser guiados para efectuar desplazamientos limitados) y estar estructurados para mover sus elementos terminales en un determinado espacio de trabajo según uno o más sistemas de coordenadas, y con un número limitado de grados de libertad. En este grupo, se encuentran los manipuladores, los Robots industriales, los Robots cartesianos y se emplean cuando es preciso abarcar una zona de trabajo relativamente amplia o alargada, actuar sobre objetos con un plano de simetría vertical o reducir el espacio ocupado en el suelo.

- **Móviles**. Son Robots con gran capacidad de desplazamiento, basados en carros o plataformas y dotados de un sistema locomotor de tipo rodante. Siguen su camino por telemando o guiándose por la información recibida de su entorno a través de sus sensores. Estos Robots aseguran el transporte de piezas de un punto a otro de una cadena de fabricación. Guiados mediante pistas materializadas a través de la radiación electromagnética de circuitos empotrados en el suelo, o a través de bandas detectadas fotoeléctricamente, pueden incluso llegar a sortear obstáculos y están dotados de un nivel relativamente elevado de inteligencia.

- **Androides**. Son los tipos de Robots que intentan reproducir total o parcialmente la forma y el comportamiento cinemático del ser humano. Actualmente, los androides son todavía dispositivos muy poco evolucionados y sin utilidad práctica, y destinados, fundamentalmente, al estudio y experimentación. Uno de los aspectos más complejos de estos Robots, y sobre el que se centra la mayoría de los trabajos, es el de la locomoción bípeda. En este caso, el principal problema es controlar dinámica y coordinadamente en el tiempo real el proceso y mantener simultáneamente el equilibrio del Robot.

- **Zoomórficos**. Los Robots zoomórficos, que considerados en sentido no restrictivo podrían incluir también a los androides, constituyen una clase caracterizada principalmente por sus sistemas de locomoción que imitan a los diversos seres vivos. A pesar de la disparidad morfológica de sus posibles sistemas de locomoción es conveniente agrupar a los Robots zoomórficos en dos categorías principales: caminadores y no caminadores. El grupo de los Robots zoomórficos no caminadores está muy poco evolucionado. Los experimentos efectuados en Japón basados en segmentos cilíndricos biselados acoplados axialmente entre sí y dotados de

un movimiento relativo de rotación. Los Robots zoomórficos caminadores multípedos son muy numerosos y están siendo objeto de experimentos en diversos laboratorios con vistas al desarrollo posterior de verdaderos vehículos terrenos, pilotados o autónomos, capaces de evolucionar en superficies muy accidentadas. Las aplicaciones de estos Robots serán interesantes en el campo de la exploración espacial y en el estudio de los volcanes.

- **Híbridos**. Corresponden a aquellos de difícil clasificación, cuya estructura se sitúa en combinación con alguna de las anteriores ya expuestas, bien sea por conjunción o por yuxtaposición. Por ejemplo, un dispositivo segmentado articulado y con ruedas, es al mismo tiempo, uno de los atributos de los Robots móviles y de los Robots zoomórficos.

Otra clasificación más generalista que se podría hacer es basada en su propósito o función. En este apartado encontraríamos:

- Industriales
- Personales/Educativos
- Militares

2.2 Robótica asistencial

2.2.1 Introducción

Debido al envejecimiento de la población se provoca una reducción de población activa, que conlleva a un aumento de personas con necesidades asistenciales. Cada vez más personas necesitan asistencia bien por envejecimiento o por necesidades especiales. En este sentido la robótica asistencial desempeña un papel importante ya que permitirá que dichas personas sean independientes y desarrollen una vida de calidad.

El campo de la robótica asistencial está creciendo, pero aún no se ha definido adecuadamente. Se ha prestado una atención significativa, debido a los grandes progresos realizados en la robótica asistencial de contacto, pero actualmente no hay una definición clara de los robots que proporcionan asistencia a través de la interacción sin contacto, cuyo nombre es robótica asistencial social. Entonces, a lo largo de este apartado se intentará aclarar los diferentes conceptos mencionados anteriormente.

2.2.2 Robótica asistencial

En el pasado, la robótica asistencial (Assistive Robotics, AR) se ha referido principalmente a los robots que ayudan a las personas con discapacidades físicas a través de la interacción física. Esta definición ya no es apropiada, puesto que ahora su alcance es mayor, ya que no cubre los robots de asistencia que ayudan a través de la interacción sin contacto, tales como los que interactúan con los pacientes convalecientes en un hospital o de la tercera edad en un hogar de ancianos.

Una definición adecuada de un robot asistencial es aquel que da ayuda y apoyo a un usuario humano. La robótica asistencial también consiste en poner al servicio de la sociedad soluciones tecnológicas y científicas para la mejora de robots de servicio, diseño de sistemas de control cooperativo (quirúrgicos), monitorización de usuarios (exoesqueletos), interfaces de control multimodal y quizás lo más importante, la integración en un entorno doméstico y asistencial bajo la forma de sistema robotizado para la asistencia de personas con necesidades especiales.

2.2.3 Robótica interactiva social

El término de robótica interactiva social (Socially Interacive Robotics, SIR) lo utilizó por primera vez Fong [17] para describir los robots cuya principal tarea era algún tipo de interacción. El término fue introducido para distinguir la interacción social de la teleoperación en la interacción humano-robot. Fong llevó a cabo un estudio de robots interactivos sociales y los evaluó según los principios de interacción social, por la categorización de los aspectos de la interacción social (voz, gestos, etc.) que utilizaban.

Las inquietudes con respecto a la percepción humana de la robótica, en particular la diferencia en la sofisticación social entre humanos y robots sociales, se abordaron, y los estudios de campo, la evaluación y la interacción a largo plazo fueron observados como áreas dignas de investigación futura.

2.2.4 Robótica asistencial social

Se define la robótica asistencial social (Socially Assistive Robotics, SAR) como una intersección entre AR y SIR. SAR comparte con la robótica asistencial el objetivo de proporcionar asistencia a los usuarios humanos, pero especifica que la asistencia es a través interacción social. Debido al énfasis en la interacción social, la SAR tiene un enfoque similar a la

SIR. En SIR, el objetivo del robot es desarrollar interacciones estrechas y eficaces con los humanos en aras de la interacción en sí misma. Por lo contrario en SAR, el objetivo del robot es crear una interacción estrecha y eficaz con un usuario humano con el propósito de dar asistencia y realizar progresos mensurables en la convalecencia, rehabilitación, aprendizaje, etc.

La motivación para definir la SAR no es crear un cisma dentro de la SIR, sino más bien para ampliar la robótica asistencial e incluir robots que operan a través de la interacción social y también entender mejor los desafíos clave de este campo cada vez mayor. También hay una amplia motivación para investigar la SAR ya que existe una multitud de tareas importantes de asistencia donde la interacción social, en lugar del contacto con el usuario, es el foco principal. Un ejemplo de esto es el estudio de la recuperación post-ictus. Una variedad de robots manipuladores asistenciales se han desarrollado para la rehabilitación posterior al accidente cerebrovascular, que mueven físicamente las extremidades del paciente como una forma de terapia física. Sin embargo, la terapia de restricción inducida (constraint induced, CI) [18], donde el terapeuta recuerda y entrena a un paciente con ictus a usar repetidamente los miembros afectados, se reconoce como uno de los métodos de rehabilitación más eficaces. Este enfoque no implica ningún contacto físico entre el paciente y el terapeuta, por tanto presenta una excelente solución para un robot asistencial social, ya que se ha demostrado que mediante el uso de sus propios miembros, los pacientes aprenden habilidades más generales y tienden a ejercer patrones de comportamiento más útiles.

Otra motivación importante para la SAR es la significativa disminución del riesgo de seguridad en la interacción humano-robot sin contacto. Debido a esta característica, los sistemas se prueban y se despliegan más fácilmente.

2.3 Tipos de programación de robots asistenciales

Antes de enunciar algunos tipos de programación, primeramente hay que definir este concepto.

Se conoce como programación a los pasos que se abordan para crear el código fuente de un programa o aplicación informática. De acuerdo a estos pasos, el código se escribe, se prueba y se perfecciona. El código fuente se basa en un conjunto de instrucciones que sigue el ordenador para ejecutar un programa. Estas instrucciones se encuentran escritas en lenguaje de programación que después son traducidas a un lenguaje máquina, que puede ser interpretado y ejecutado por el hardware del equipo.

En concreto, en la robótica, la programación hace referencia al proceso que indica la secuencia de acciones a llevar a cabo por el robot durante la ejecución de una tarea determinada.

2.3.1 Programación por demostración

La programación por demostración (Programming by demonstration, PbD) es una técnica, donde, al contrario de la programación detallada que convencionalmente se hace al robot, este aprende movimientos parciales o incluso una tarea completa, con solo ver una o varias demostraciones de los movimientos o la tarea [1]. Es decir, se hace uso de una demostración directa o personal.

El aprendizaje por demostración inicialmente, fue abordado en trabajos de psicología y neurociencia, mostrándolo como la forma natural en la que los humanos [2] y algunos animales [3] aprenden. Es por tanto, una técnica poderosa en la adquisición de conocimiento que, aplicada en robótica, permitiría la programación de robots con la habilidad de aprender comportamientos complejos e interactuar inteligentemente con el ambiente. Así mismo, abre la posibilidad de que varios robots puedan ser programados simultáneamente, aun cuando los robots difieran morfológicamente del demostrador o maestro (que puede ser un humano u otro robot). Los trabajos previos abarcan aplicaciones que van desde la réplica exacta de la trayectoria seguida por un efector final, hasta la imitación de gestos faciales o agarres con una o dos manos.

Los sub-problemas involucrados en la imitación son: observación, reconocimiento y ejecución de la imitación. La mayor parte de las investigaciones se centran en uno o más de estos sub-problemas, ninguna los cubre todos. Algunos de los sub-problemas abordados por todos son: la segmentación, el procesamiento de la información relevante que acompaña la acción demostrada y la elección de una representación apropiada. Dentro de los sub-problemas sin resolver se destacan la elección de un maestro apropiado, la selección automática del momento para realizar una acción aprendida y la evaluación cuantitativa de la imitación cuando está orientada a alcanzar una meta y no una réplica exacta de las acciones ejecutadas por el demostrador.

2.3.1.1 Concepto de imitación

La imitación es un concepto difícil de definir, y no se ha adoptado una definición estándar. De hecho, parece adaptarse a la conveniencia de cada trabajo de ingeniería. De acuerdo con la Real Academia de la Lengua Española, imitar se define como:

i. “Ejecutar algo a ejemplo o semejanza de otra cosa”;

ii. “Dicho de una cosa: Parecerse, asemejarse a otra”;

iii. “Hacer o esforzarse por hacer algo lo mismo que otro o según el estilo de otro”.

La imitación implica reconocer y reproducir las acciones de otro, concentrándose en los movimientos destacados y descartar detalles motrices con poca importancia. Esta habilidad es importante cuando las demostraciones son realizadas por humanos, quienes presentan una alta variabilidad en las ejecuciones al realizar una tarea, por lo que no existe algún límite o umbral absoluto para definir el éxito en la imitación, aparte de fallar en la asociación entre comportamientos.

2.3.1.2 Concepto de demostración

La demostración es un modelo de exposición pero más lógica y concreta, la cual tiende a confirmar un resultado anteriormente enunciado. Es decir, una demostración confirma e ilustra explicaciones teóricas y también da un esquema correcto y seguro para ejecutar una tarea.

La demostración tiene ciertos objetivos:

- Adquirir las destrezas de manipulación básicas que la operación exige.
- Aprender a manejar el equipo que será utilizado en la operación a ejecutar.
- Iniciar el estudio de la operación de modo concreto.

Hay varios tipos de demostración:

- **Directa o personal**: es la realizada por el propio instructor. Exige del instructor planificación y técnica, así como la eventual ayuda de material audiovisual a efectos de facilitar la presentación de la operación.
- **Sustitutiva:** es la efectuada por monitores o asistentes bajo control del instructor.
- **Indirecta:** es la realizada por medios audiovisuales u otros recursos semejantes. Se complementa con observaciones y explicaciones del instructor. Este tipo de demostración despierta sumo interés e ilustra de manera realista los pasos de la operación en cámara lenta, por ejemplo permite un análisis más detallado y exacto de los movimientos ejecutados por el instructor.

2.3.1.3 Sistema de aprendizaje por demostración

Para definir los bloques constitutivos del sistema robótico de aprendizaje por demostración se adopta la definición dada por [4]: “imitación es cuando un gesto es observado, reconocido y ejecutado”.

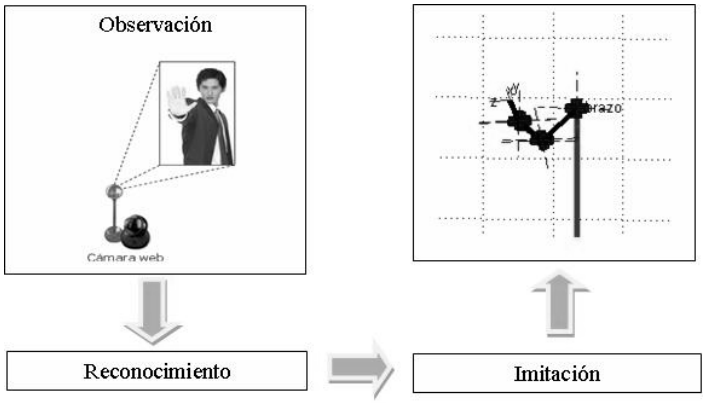

Figura 5. Diagrama de bloques general del sistema

El bloque de "Observación" extrae la información visual (vídeos) del ambiente mediante una cámara Web con baja resolución y condiciones ambientales semi-controladas.

El bloque de "Reconocimiento" genera, a partir de los vídeos una representación que reúne la información de movimiento a lo largo del tiempo [5]. En este bloque, la codificación de la información de movimiento instantánea utiliza técnicas bio-inspiradas en la forma en la que se procesa la información de movimiento en el cerebro de macacos, similar al cerebro humano. Esta representación de la información facilita el proceso de reconocimiento del gesto efectuado por el demostrador y su posterior imitación.

El bloque de "Imitación" relaciona la representación de la información visual con el espacio motor, generando de esta manera un mapa visuo-motor. Gracias a este mapa, se dota al sistema de una transformación directa a variables articulares del robot, que permiten que el robot realice su propia interpretación de la escena observada. En [6], se describe la manera de obtener los valores articulares desde una imagen monocular tomada por una cámara Web sin calibrar, de tal forma que un brazo robótico con 6 grados de libertad (GDL) rotacionales pueda imitar la pose de un brazo humano durante la ejecución de un gesto.

2.3.1.4 Trabajos relacionados

El manejo de herramientas por parte de un robot, se puede dividir en subtareas como reconocimiento de la herramienta, agarre y posicionado de la herramienta, o reproducción de las acciones de manipulación de la herramienta para realizar la meta de la tarea. Estas subtareas, pueden ser aprendidas por demostración.

En [7] J. G. Hoyos-Gutiérrez y otros, presentan una técnica que permite llevar una herramienta hasta la cabeza de un tornillo siguiendo una trayectoria similar a las demostradas por un ser humano. Se utiliza un robot humanoide NAO, véase Figura 6, con capacidad de

visión. Empleando técnicas de visión por ordenador y modelos probabilísticos, se logra flexibilidad en la reproducción de la tarea, ya que esta permite alinear la llave con la cabeza del tornillo, aún si se presentan variaciones en la ubicación y orientación de la cabeza del tornillo. Esta información visual es una de las entradas a un modelo probabilístico de las trayectorias, con el que se logra generar una nueva trayectoria.

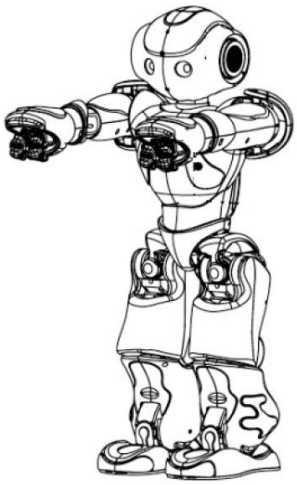

Figura 6. Robot NAO

El robot NAO, es un robot humanoide programable de 58 centímetros de alto, desarrollado por la compañía francesa Aldebaran. Entre sus principales características se encuentran: un cuerpo con 25 grados de libertad y una red de sensores, incluyendo dos cámaras RGB. Aunque se utilizó un robot humanoide, la técnica puede ser adaptada a robots industriales y la contribución consiste en la aplicación de la técnica de modelos probabilísticos en el manejo de herramientas.

Por otra parte, en 2005 Zollner y otros [8], a partir de una sola demostración de una tarea compleja, generan un programa de esta. Es decir, realizan un mapeo de la tarea para su posterior extrapolación y generación de un programa. Su ventaja es que requiere una sola demostración, y como desventajas están el que requiere la definición de reglas heurísticas para poder generalizar posiciones, esto último solo se hace para cambios de posición y forma de los objetos, y no para la orientación.

Otro método [9] expuesto en 2010 por Kruger y otros, presenta una técnica que detecta primitivas de movimiento de manera no supervisada y con ellas sintetizan tareas, es decir, usa un modelo de reconocimiento de gestos para sintetizar las tareas de movimiento. Las primitivas son reconocidas y modeladas usando modelos ocultos de Markov paramétricos (PHMM por sus siglas en inglés), además para la detección de las primitivas, se valen del estado de los objetos. Plantean una dualidad entre el movimiento de los objetos y las acciones realizadas por el humano, basados en esto, usan la segmentación del movimiento de los

objetos para separar las acciones del humano en primitivas. Los autores no ahondan en el tema de la obtención de la gramática que se genera a partir de la tarea.

En 2012 Niekum y otros [11] plantean una técnica para resolver tareas complejas. Primero segmentan la tarea con BP-ARHMM (del inglés Beta Process Auto Regresive Hidden Markov Models). Luego las subtareas son modeladas usando primitivas de movimiento dinámico (DMP) y vinculadas a un marco de referencia. Una mejora al anterior, fue presentada por los mismos autores en 2013 [12], donde emplearon un autómata de estados finitos para la tarea de enroscar la pata a una mesa. A través de aprendizajes interactivos por parte del ser humano, puede aprender de manera incremental a tomar la pata de la mesa de otras ubicaciones y orientaciones, esto implica adicionar ramas al autómata de estados finitos, las cuales luego son condensadas con las ramas iniciales en sus tramos similares. Las anteriores técnicas, pueden generalizar en cada subtarea, un punto inicial o final, a diferencia de la técnica aquí propuesta que al emplear TPGMM puede tener múltiples parámetros de la tarea.

Cubek y otros en 2015 [15], presentaron una técnica que realiza la tarea de tomar y poner múltiples objetos, esta aprende relaciones conceptuales de forma y color de los objetos con acciones a realizar. Aunque puede generalizar para nuevos objetos, no modelan las trayectorias demostradas, con lo cual no es posible que el robot pueda realizar movimientos que requieran cierta orientación.

2.3.2 Programación por guiado o refuerzo de aprendizaje

La programación por guiado o por refuerzo de aprendizaje consiste en llevar al robot, o a una maqueta del mismo, por el camino que se desea que repita posteriormente, registrando el propio robot las configuraciones adoptadas a lo largo de la trayectoria.

2.3.2.1 Modos de guiado

Dentro de esta programación hay distintos modos de guiado. Uno es el guiado pasivo, donde los actuadores del robot están desconectados y el programador aporta de forma directa la energía para mover el robot. Y otro es el guiado activo, en el cual se emplea el propio sistema de accionamiento del robot, controlado desde una botonera o un joystick para que sea este el que mueva sus articulaciones.

Dentro del modo pasivo hay dos tipos:

- **Pasivo directo**, donde el programador mueve directamente el extremo del robot. Y la unidad de control registra posiciones de forma automática, realizando un muestreo de la trayectoria con un intervalo de tiempo determinado.
- **Pasivo indirecto o por maniquí,** en el que en vez del robot real se mueve un maniquí con su misma configuración cinemática pero mucho más ligero y fácil de mover. Por tanto el maniquí necesitará los sensores necesarios para leer las posiciones de las articulaciones.

Pasando al modo de guiado activo, se pueden encontrar dos grupos que se diferencian en la potencia del sistema robótico.

- **Básico**. El robot es guiado por los puntos por los cuales se desea que pase durante la fase de ejecución automática del programa.
- **Extendido**. Permite especificar, junto a los puntos por los que deberá pasar el robot, datos relativos a la velocidad, tipo de trayectoria, precisión con la que se quiere alcanzar los puntos, control del flujo del programa, atención a entradas/salidas binarias, etc.

2.3.2.2 Inconvenientes

Este tipo de programación tiene ciertos inconvenientes. Para empezar, necesita disponer del robot y su entorno durante la programación de la aplicación, lo que conlleva un elevado coste económico y además el riesgo de accidentes es mayor. Segundo, también necesita parar la instalación para cambiar la aplicación a ejecutar. Tercero, normalmente no se genera documentación escrita y comentada de los programas realizados, lo que puede ocasionar problemas en futuras revisiones o actualizaciones. Y cuarto se dificulta la depuración de grandes programas.

2.3.2.3 Trabajos relacionados

En 2011 T. Tamei y otros [25], exponen un estudio realizado cuyo objetivo es prestar asistencia robótica en la acción de vestirse, ya que es todavía un campo abierto para la robótica a pesar de que es una de las actividades de asistencia más básicas e importantes en la vida diaria de las personas mayores, así como personas con discapacidad. La asistencia al vestirse es un problema difícil de resolver ya que los robots deben interactuar con ropa, material no rígido generalmente representado en un espacio multidimensional, y con la

persona asistida cuya postura puede variar durante la asistencia. Por lo tanto, se requiere que el robot pueda llevar a cabo la difícil gestión de dos tareas en la asistencia al vestirse: la manipulación de materiales no rígidos y la adaptación de los movimientos que ayudan a la postura de la persona asistida. Para superar estas dificultades, proponen utilizar el aprendizaje por refuerzo unido al conocimiento del estado de la ropa representado en coordenadas de topología.

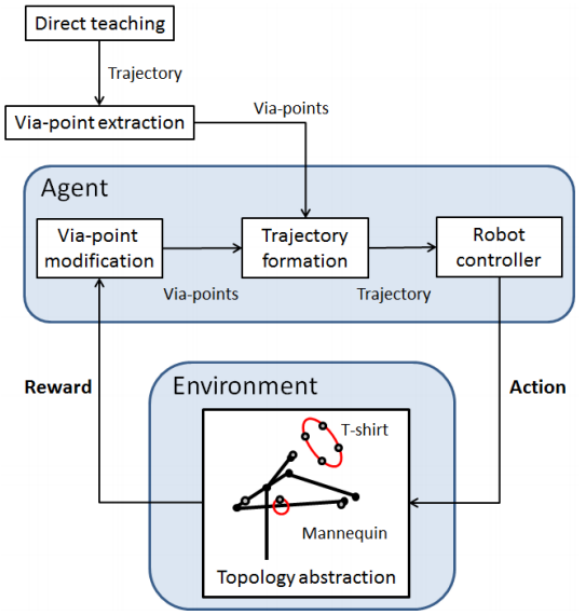

Figura 7. Esquema de aprendizaje en la asistencia al vestirse

En 2014 Martínez y otros [14], emplean una gramática propia y aprendizaje por refuerzo para una tarea de ensamble, consiguiendo un aprendizaje activo de secuencias de manipulación. Con el movimiento y contacto de los objetos a ensamblar construyen unos grafos, con los cuales entrenan un algoritmo de aprendizaje por refuerzo relacional, según las precondiciones y efectos deseados más adecuados, el algoritmo decide que acción ejecutar. Su ventaja está en que es tolerante a variaciones de la percepción de los movimientos, partes no identificadas y errores de inserción.

2.3.3 Programación textual

La programación textual permite indicar la tarea al robot a través de un lenguaje de programación específico. En la programación textual se escriben programas que tienen una

secuencia de órdenes que son editadas y escritas por el usuario y posteriormente ejecutadas por el robot.

Un programa se puede procesar de diferentes modos: Interpretado, que facilita la depuración y puesta a punto. Y compilado, necesario en lenguajes con sintaxis muy compleja, aumenta la velocidad de ejecución.

2.3.3.1 Clasificación

La programación textual se puede clasificar en tres niveles, según la potencia de las órdenes:

- **Nivel robot**, si las instrucciones especifican cada uno de los movimientos que ha de realizar el robot, como velocidad, direcciones de aproximación y salida, apertura y cierre de la pinza, etc. Es decir, si las órdenes hacen referencia a las acciones que debe ejecutar el robot.
- **Nivel objeto**, si las instrucciones se refieren al estado en que deben ir quedando los objetos manipulados. Disminuye la complejidad del programa, y la programación se realiza de manera más cómoda ya que las instrucciones se dan en función de los objetos a manejar.
- **Nivel tarea**, si las instrucciones se refieren al objetivo a conseguir. Con esto, el programa se reduce a una única sentencia, ya que se especifica qué es lo que debe hacer el robot en lugar de cómo debe hacerlo.

3 DESARROLLO DEL TRABAJO

El principal objetivo de este capítulo es relatar los pasos seguidos en el desarrollo de la demostración de una terapia de rehabilitación motora de alcance, a través de una brazo robótico. Se realizará un análisis más concreto de las tecnologías utilizadas y se explicarán las decisiones más importantes que se han tomado, con la única finalidad de cumplir todos los objetivos marcados al inicio de este trabajo.

3.1 Visión general de la solución

Antes de proceder a detallar la configuración y las herramientas utilizadas, es necesario mostrar, en primer lugar, una visión general de la tarea que se desea desarrollar. La tarea consiste en programar una demostración de una terapia de rehabilitación motora de alcance, que se realizará en simulación a través de la interfaz web de ASIBOT y el simulador OpenRAVE.

El ejercicio de rehabilitación consiste en que el brazo robótico realizará ciertas trayectorias y el último eslabón o articulación del mismo se posicionará de tal manera que el paciente con su mano intentará llegar a dicha posición. Tras la realización de estos movimientos repetidamente, se consigue que el paciente vaya mejorando la movilidad de su extremidad; por consiguiente se estaría realizando una terapia de rehabilitación con una interacción humano-robot sin necesidad de la presencia del médico supervisor de la terapia. Pero esto no quiere decir que no se necesite a un médico, ya que él es quien valora la evolución de la rehabilitación; además de haber especificado previamente qué tipo de trayectorias o alcances debe seguir el robot, dependiendo del paciente y la evolución del mismo.

Con la solución aportada en este trabajo se quiere lograr que la interacción humano-robot sea más natural y sencilla para usuarios no experimentados, que en este caso son los terapeutas de la rehabilitación, pero fuera de este ámbito puede ser cualquier persona que necesite de una asistencia.

3.2 Habilitación de la interfaz web de ASIBOT

Antes de explicar los pasos seguidos para habilitar la interfaz web se procede a definir tanto que es una interfaz web y la tecnología que va detrás de ella, como la estructura o arquitectura de la misma.

Como se mencionó anteriormente, esta interfaz web servirá de herramienta a los terapeutas para elaborar los ejercicios de alcance en la terapia de rehabilitación con el robot ASIBOT.

3.2.1 Interfaz web

En el ámbito de sitios web, se denomina interfaz al conjunto de elementos de la pantalla que permiten al usuario realizar acciones sobre el sitio web que está visitando. Es decir, es un medio de comunicación entre el usuario y un dispositivo, máquina o equipo, que haga dicha comunicación amigable e intuitiva.

El objetivo de realizar una interfaz web del brazo robótico ASIBOT es incrementar la aceptación de una solución software para el control y manejo de dispositivos, acercando la robótica al usuario final [24].

La tecnología usada para el desarrollo de la interfaz web se basa en HTTP, que es el protocolo más utilizado para la transmisión de páginas web, e implementaciones del concepto de “Cloud Computing” para la interacción humano-robot.

3.2.2 Tecnologías y herramientas utilizadas

En este apartado se realiza una descripción de los programas software empleados mediante los cuales se ha conseguido la arquitectura de la interfaz web de ASIBOT que se expone en este trabajo. Todos los programas software necesarios se detallan a continuación.

3.2.2.1 Computación en la nube

El término “Cloud Computing”, traducido en general como “computación en la nube”, fue acuñado por primera vez en 1997 por Ramnath Chellappa. El término tiene su origen en la forma de nube con la que se suele representar una red de computadores, Figura 8, que a su vez se utilizaba anteriormente en el contexto de redes de telefonía. En estos diagramas, el

perímetro de la nube delimita la barrera que linda entre las responsabilidades del cliente (en el modelo clásico cliente-servidor) y el proveedor de servicios de Internet (ISP, Internet Service Provider). El concepto de la "nube" extiende este perímetro, abarcando innumerables servidores, ISP, y toda la infraestructura de red que los conecta.

La importancia de este paradigma reside en que la "nube" puede ofrecer un número virtualmente ilimitado de servicios, de forma descentralizada y a la vez integrada. Estos servicios son accesibles desde cualquier dispositivo con capacidades de navegación web (smartphones, PDAs, ebooks, netbooks, y tablet PCs entre otros) sin necesidad de instalación de software adicional. Las funcionalidades de éstos pueden ser tales que lleguen a reemplazar por completo a los paquetes de software que se deben instalar. Un ejemplo de servicio ampliamente utilizado en la actualidad es Gmail, que puede utilizarse en lugar de aplicaciones como Thunderbird o Outlook. Existen también soluciones para empresas, como los servicios de planificación de recursos basados en web ofrecidos por Salesforce o SAP AG [23].

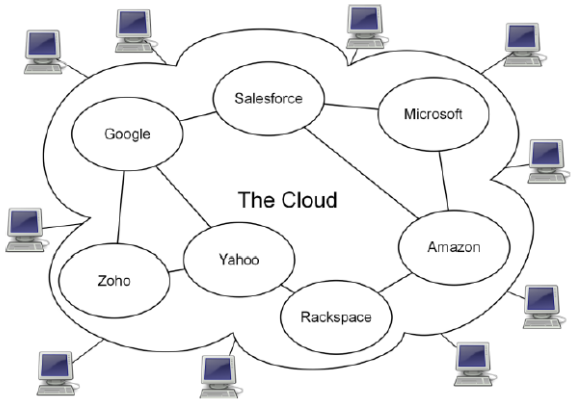

Figura 8. La forma de nube engloba servidores, ISP y toda la infraestructura.

3.2.2.2 Tecnología en servidores HTTP

El protocolo de transferencia de hipertexto (HTTP, Hypertext Transfer Protocol) es, por excelencia, el protocolo más empleado por los servidores de páginas web para transmitir éstas a sus clientes. Las arquitecturas y los sistemas operativos de estas máquinas puede variar, pero a efectos prácticos, todas ellas realizan esta misma funcionalidad (servir páginas web) y utilizan para ello una aplicación común: un servidor de HTTP. El protocolo HTTP permite la transmisión de páginas web escritas en HTML puro, además de la transmisión de imágenes, hojas de estilo, y de contenido más dinámico como pueden ser componentes de ActiveX, JavaScript o Applets de Java (estos últimos tres deben ser computados por el lado cliente). Existen diversas

aplicaciones y tecnologías en los que se puede basar un servidor de HTTP, y a continuación exponemos algunas de las soluciones de uso más extendido.

- Apache HTTP Server ("httpd"): Se trata de la aplicación servidor de HTTP más utilizada en el mundo, con un 62,71% de cuota de mercado a Mayo de 2011 (Netcraft, 2011). En principio, se trata de una simple aplicación que se ejecuta y que funciona de servidor de HTTP a medida que llegan peticiones por parte de clientes (dispositivos con una conexión TCP/IP que los enlaza con la máquina). Sin embargo, su arquitectura modular permite la ampliación de las funcionalidades de su núcleo. Módulos como mod_ssl le aportan posibilidades de servir HTTP bajo conexión segura (HTTPS); mod_python permite integrar un intérprete del lenguaje Python dentro del servidor.
- Alternativas semejantes a Apache HTTP Server: Existe en el mercado un gran número de alternativas a Apache HTTP Server, de licencia tanto libre como propietaria. Entre ellos se encuentran los servidores IIS de Microsoft (con una cuota de mercado del 18,37% según las mismas estadísticas), ngnix de Igor Sysoev (7.35%), GWS de Google (5.00%), o lighttpd (0.58%).
- Infraestructuras para la creación de aplicaciones web: Se trata de librerías e infraestructuras software para desarrolladores, realizados para diferentes lenguajes de programación (C++, Java, Pearl, Python, Ruby…), y orientadas al desarrollo de webs dinámicas y aplicaciones basadas en web. Su grado de complejidad es variado, siendo en general mayor a medida que el desarrollador utiliza una infraestructura que le permite mayor flexibilidad en cuanto al formato de las páginas ofrecidas y la potencia a la hora de integrar la interfaz con componentes de más bajo nivel.

3.2.2.3 CMake

CMake[1] es una herramienta multiplataforma de generación o automatización de código. Diseñada para construir, probar y empaquetar software, se utiliza para controlar el proceso de compilación del software usando ficheros de configuración sencillos e independientes de la plataforma.

CMake genera espacios de trabajo que pueden usarse en el entorno de desarrollo deseado y soporta la generación de ficheros para los sistemas operativos Linux, Windows y Mac OS X. Esta característica facilita el mantenimiento y elimina la necesidad de tener varios conjuntos de ficheros para cada plataforma.

[1] https://www.cmake.org/ (última revisión 7/9/2016)

3.2.2.4 KDL (Kinematics and Dynamics Library)

KDL[2] es una librería dedicada a la cinemática y la dinámica, distribuida por el proyecto Orocos[3]. Desarrolla un marco de aplicación independiente para el modelado y el cálculo de cadenas cinemáticas, tales como robots, modelos biomecánicos humanos, figuras animadas por ordenador, herramientas computacionales, etc.

Además, proporciona librerías de clases para objetos geométricos que ofrecen un excelente soporte para trabajar con vectores, puntos y sistemas de referencia, así como con cadenas cinemáticas de diversos tipos (serie, humanoide, paralelo o móvil) y sus especificaciones de movimiento e interpolación.

3.2.2.5 YARP (Yet Another Robot Platform)

En cuanto a la comunicación se utiliza La librería YARP[4] que se describe como una arquitectura software de código abierto para robots que ayuda a organizar la comunicación entre los sensores, procesadores y actuadores, permitiendo un acoplamiento flexible entre ellos.

Incluye un modelo de comunicación que permite un transporte neutral, de modo que el flujo de datos se desacopla de las características intrínsecas de las redes subyacentes y protocolos en uso.

YARP está diseñada para operar conjuntamente con otras arquitecturas, lo que resulta importante para su longevidad a largo plazo. El uso de YARP facilita el intercambio de código entre investigadores, principalmente cuando el tiempo necesario para desarrollar una plataforma robótica y usarla para investigación es un factor crítico.

YARP fue inicialmente desarrollada por el MIT y LiraLab (Universidad de Génova) para plataformas robóticas humanoides y cuenta también con el soporte del Instituto Italiano de Tecnología (IIT). Presenta como características principales ser una librería liviana y fácil de usar para los principiantes, a la vez que dispone de un código complejo y especialmente eficiente para ser utilizado por los usuarios más expertos. YARP es compatible con Linux, Windows y Mac OS X y depende exclusivamente de ACE para su implementación.

La interfaz nativa de YARP está basada en el lenguaje de programación orientado a objetos C++. No obstante, mediante la herramienta de libre distribución denominada SWIG[5] es posible

[2] http://www.orocos.org/kdl (última revisión 7/9/2016)
[3] http://www.orocos.org/ (última revisión 7/9/2016)
[4] http://wiki.icub.org/wiki/YARP (última revisión 7/9/2016)
[5] http://www.swig.org/ (última revisión 7/9/2016)

generar *wrappers* para el código fuente en C++, lo que permite su utilización con distintos lenguajes de programación como Python, Java, Matlab (vía Java), TCL, Lisp, C#, etc.

La comunicación mediante YARP generalmente sigue el patrón de diseño *Observer*. Dispone de objetos especiales, denominados puertos, que entregan mensajes a cualquier número de observadores (otros puertos), empleando cualquier número de procesos distribuidos a través de diferentes máquinas físicas. Dichos procesos pueden incluso ser ejecutados bajo distintos sistemas operativos y ser implementados utilizando distintos lenguajes de programación, observar Figura 9.

YARP soporta el uso de los siguientes tipos de protocolo de comunicación, lo que permite un mejor aprovechamiento de las cualidades más destacadas de cada uno de ellos:

- TCP/IP: protocolo fiable que puede ser utilizado para garantizar la recepción de los mensajes enviados.

- UDP: es un protocolo más rápido que el TCP/IP, utilizado para conexiones punto a punto, pero que no ofrece garantías en la recepción de mensajes.

- Multicast: protocolo utilizado para crear conexiones entre múltiples puntos; resulta eficaz para distribuir la misma información a un gran número de objetivos, por ejemplo, las imágenes de una cámara.

- Memoria compartida: empleado para conexiones locales. Este protocolo se selecciona automáticamente siempre que sea posible, sin la necesidad de ninguna intervención por parte del programador.

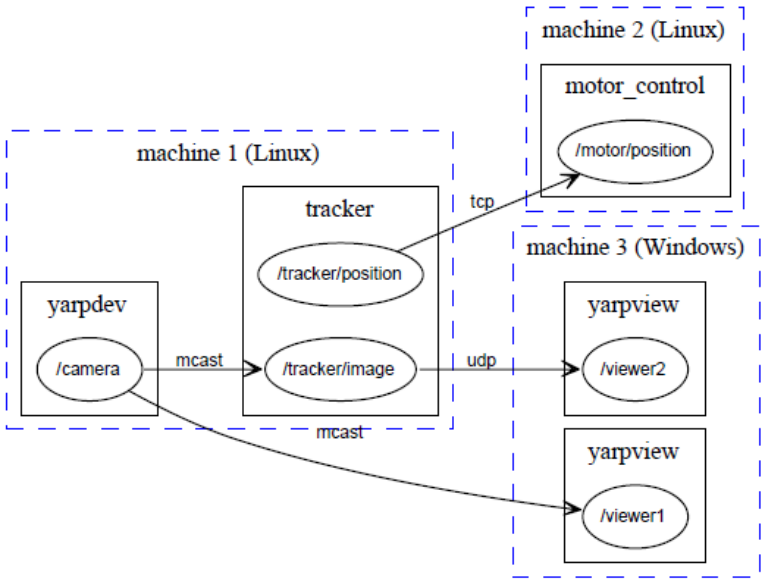

Figura 9. Ejemplo de una red de puertos en YARP con protocolos de comunicación diferentes.

YARP presenta herramientas útiles aplicadas a la línea de comandos para realizar operaciones con la red de puertos, así como utilidades gráficas y basadas en la web.

3.2.3 Arquitectura software

Como se ha mencionado en la sección 3.2.2.2, la plataforma más usada para servir contenido web es Apache HTTP Server, en la que los clientes realizan peticiones al servidor y éste transmite los contenidos web a través del protocolo HTTP. Este modo de funcionamiento tiene un límite en el diseño y en alcance de una implementación, ya que, lo que se realiza dentro de la aplicación (Apache) queda dentro de la misma, es decir, queda encapsulada de alguna forma.

Por tanto, para superar esta limitación se ha utilizado una infraestructura para la creación de aplicaciones web, que sea compatible con el lenguaje de programación Python, ya que se trata de un lenguaje que combina los beneficios de un lenguaje interpretado (rapidez en desarrollo y depuración) y una alta eficiencia computacional. La librería seleccionada para servir HTTP ha sido CherryPy, debido a su sencillez y claridad que aporta su API.

La arquitectura software utilizada para las comunicaciones con el robot es YARP. Su elección se debe a su naturaleza ligera y multiplataforma, su API sencilla y compatibilidad con diversos lenguajes. Muchas de las librerías que se autodenominan “infraestructuras” imponen una estructura de programa, incluso eliminando la posibilidad de programar una función main() con contenido propio. YARP sin embargo, se utiliza mediante sencillas llamadas a funciones dentro del código fuente, en este caso principalmente orientado a la infraestructura dada por CherryPy. Este mecanismo de integración entre la interfaz web y el robot a controlar está representado de forma esquemática en la Figura 10.

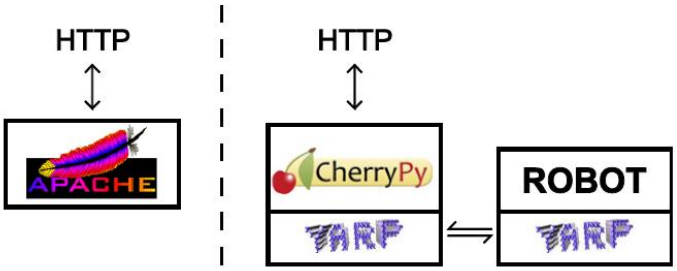

Figura 10. Comparación de una arquitectura software normal (izq.) con la adoptada en este caso (dcha.)

3.2.4 Actualización y puesta en marcha de la interfaz web

Una vez dada la definición de interfaz web y sabiendo las tecnologías que se han usado, se procede a documentar los pasos seguidos para actualizar y poner en marcha la interfaz de ASIBOT.

Lo primero de todo es instalar los programas necesarios para después probarlos y comprobar su funcionamiento actual. Para ello se sigue el tutorial[6] de instalación en Ubuntu. Después de la instalación se comprueba que la aplicación que ejecuta el módulo de la interfaz web (*webInterface*) no está habilitada por tanto hay que habilitarla con una serie de comandos, a través de la propia consola de Ubuntu, que se presentan a continuación.

```
cd asibot-main
cd build
cmake .. -DENABLE_webInterface=ON
make
sudo make install
```

Figura 11. Comandos para habilitar el módulo de la interfaz web de ASIBOT

Después de habilitar e instalar el módulo de webInterface, se procede a probar su estado. Pero se encuentran varios errores de castings dinámicos y estáticos, y sobre todo problemas de rutas para que webInterface encuentre los recursos necesarios para su funcionamiento. Con estos errores se puede apreciar el avance que se ha llevado a cabo en la manera de acceder a ficheros dentro de la memoria de un ordenador, es decir, en el procesamiento de los ordenadores. Teniendo en cuenta estos errores y sobre todo respetando el objetivo de intentar no modificar el código fuente desarrollado en su momento, se procede a una actualización de rutas de ficheros y directorios.

Lo primero que se ha abordado, por importancia de resolución para poder continuar, es el tema de los castings, que tratan sobre la conversión o transformación de una variable primitiva de un tipo a otro tipo. En este caso se quiere acceder a variables que no son constantes de una manera en la que se las trata como tal, por tanto hay que convertir esas variables en constantes. Un ejemplo de ello es el siguiente cambio:

[6] http://robots.uc3m.es/dox-asibot-main/install_on_ubuntu.html (última revisión 7/9/2016)

```
boost::shared_ptr<SensorBase::CameraGeomData> pcamerageomdata =
boost::dynamic_pointer_cast<SensorBase::CameraGeomData>(psensorbase-
>GetSensorGeometry(SensorBase::ST_Camera));
boost::shared_ptr<SensorBase::CameraGeomData const> pcamerageomdata =
boost::dynamic_pointer_cast<SensorBase::CameraGeomData const>(psensorbase-
>GetSensorGeometry(SensorBase::ST_Camera));
```

Figura 12. Casting dinámico de una variable constante

Tal y como se puede observar en la figura anterior se ha realizado un cambio de variable no constante a constante. Cuya interpretación es la siguiente: en la parte sombreada de rojo consta lo que había antes y en la parte sombreada de azul lo que hay actualmente después de la modificación, señalizado en verde el cambio realizado.

Pasando a la actualización de rutas de ficheros y directorios se encuentra la ausencia de un directorio para la aplicación de webInterface, por tanto primero se crea el fichero CMakelists.txt, el cual contiene la configuración de cada fichero o directorio que se usa al compilar el programa, para su posterior instalación.

A continuación se puede observar una figura en la que queda constancia de cómo puede ser el aspecto de un fichero de configuración CMakeLists.txt.

```
set(appname webInterface)

file(GLOB conf ${CMAKE_CURRENT_SOURCE_DIR}/conf/*.ini)
file(GLOB html ${CMAKE_CURRENT_SOURCE_DIR}/html/*.*)
file(GLOB html/fig ${CMAKE_CURRENT_SOURCE_DIR}/html/fig/*.*)

yarp_install(FILES ${conf} DESTINATION ${ASIBOT_CONTEXTS_INSTALL_DIR}/${appname})
yarp_install(FILES ${html} DESTINATION ${ASIBOT_CONTEXTS_INSTALL_DIR}/${appname}/html)
yarp_install(FILES ${html/fig} DESTINATION ${ASIBOT_CONTEXTS_INSTALL_DIR}/${appname}/html/fig)
```

Figura 13. Aspecto del fichero CMakeLists.txt de webInterface

Teniendo el directorio de webInterface con su respectivo CMakeLists.txt, el paso siguiente es actualizar los ficheros que hacen referencia a él, para que de esta manera la aplicación de webInterface pueda encontrar los recursos (archivos) necesarios para su funcionamiento.

A continuación se van a presentar ciertas figuras donde se puede observar los diferentes cambios realizados para la habilitación y actualización de la interfaz, así como también se adjuntarán unas fotos del aspecto de la interfaz web de ASIBOT.

```
ConstString htmlPath = rf.getContextPath() + "/../html/";
printf("WebInterface using htmlPath: %s\n",htmlPath.c_str());
responder.setHtmlPath(htmlPath);
responder.setResourceFinder(rf);
ConstString userPath = rf.getContextPath() + "/../user/";
printf("WebInterface using userPath: %s\n",userPath.c_str());
responder.setUserPath(userPath);
```

Figura 14. Código actualizado del archivo WebInterface.cpp

```
bool WebResponder::setHtmlPath(const ConstString& _htmlPath) {
    htmlPath = _htmlPath;
bool WebResponder::setResourceFinder(ResourceFinder &rf) {
    this->rf = rf;
    return true;
```

Figura 15. Código actualizado del archivo WebResponder.cpp

```
string WebResponder::readHtml(const ConstString& fileName) {
    ConstString filePath = htmlPath + fileName;
    ConstString filePath = rf.findFileByName(std::string("html/")+fileName);
    return readFile(filePath);
```

Figura 16. Código actualizado del archivo WebResponder.cpp

```
class WebResponder : public PortReader {
protected:
    ResourceFinder rf;
    bool simConnected, realConnected;
    ConstString htmlPath;
    ConstString userPath;
    ConstString resourcePath;
    string readFile(const ConstString& filePath);  // needs
@ -73,7 +73,7 @@ class WebResponder : public PortReader {
    bool init();
    bool closeDevices();
    bool read(ConnectionReader& in);
    bool setHtmlPath(const ConstString& _htmlPath);
    bool setResourceFinder(ResourceFinder &rf);
    bool setUserPath(const ConstString& _userPath);
    bool setResourcePath(const ConstString& _resourcePath);
};
```

Figura 17. Código actualizado del archivo WebResponder.h

Antes de ver el aspecto de la interfaz web de ASIBOT, se debe mencionar los pasos a seguir para lograr lanzar instancia de webInterface.

Primero, tal y como se ha mencionado en el apartado de arquitectura software se debe instanciar un servidor YARP que sirva de enlace de comunicación entre el servidor http, es decir la interfaz web, y el simulador o el robot (aunque todavía no se ha hablado del simulador, ya que se verá en el siguiente apartado de manejo del simulador).

Y por último, después de tener el servidor de YARP listo, se instancia webInterface desde otra consola. Consiguiendo de esta manera que quede habilitada la interfaz web para su apertura a través de un navegador web.

Figura 18. Lanzamiento de una instancia de yarp server

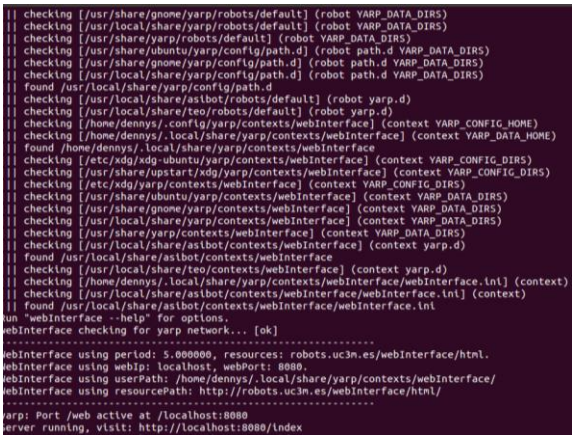

Figura 19. Lanzamiento de una instancia de webInterface

Una vez activos yarp y webInterface, se puede ir a un buscador web e introducir la dirección de la interfaz (http://localhost:8080/index), que tiene el siguiente aspecto:

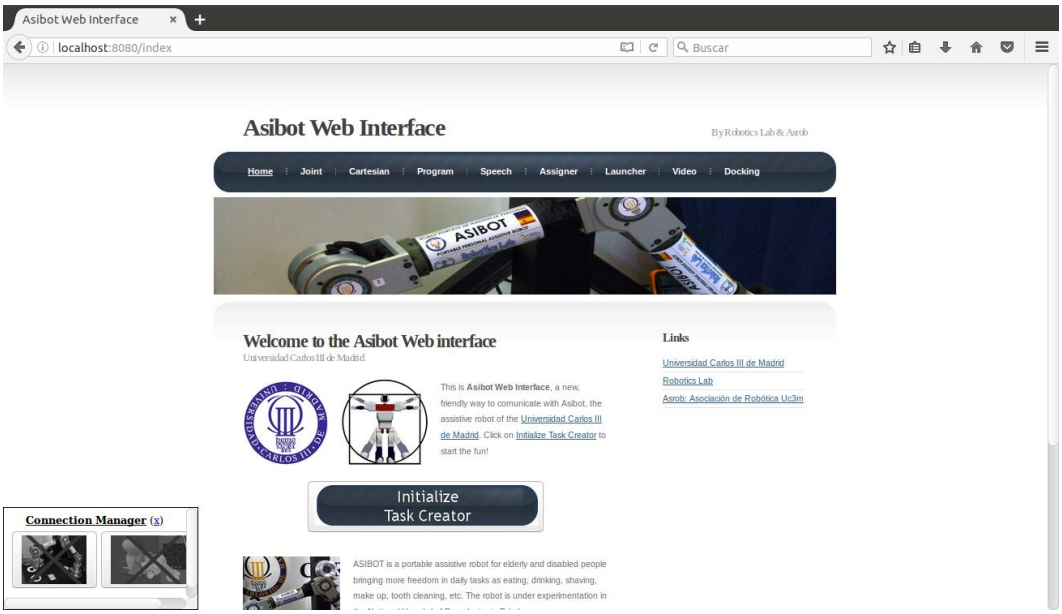

Figura 20. Interfaz web de ASIBOT

Tal y como se puede observar en la figura 20, la interfaz web tiene 9 pestañas (Home, Joint, Cartesian, Program, Speech, Assigner, Launcher, Video y Docking). Un gestor de conexiones (connection manager) que permite establecer y terminar la comunicación con el robot real y el simulador, y que está presentado en la esquina inferior izquierda de la ventana del navegador.

3.3 Manejo del simulador, mostrando la comunicación entre la interfaz web y el simulador

En este apartado se expondrá más en profundidad acerca del simulador utilizado para el desarrollo del trabajo.

Cabe destacar que un entorno de simulación es crucial para una fase de desarrollo, que permita practicar con un brazo robótico asistencial antes de su uso en el entorno físico real. Por este motivo se ha utilizado el simulador OpenRAVE, dado que es ligero y modular.

Además del uso de un entorno de simulación, se va a realizar una comunicación entre la interfaz web, anteriormente habilitada, y el propio simulador para poder realizar los movimientos del robot a través de una interfaz más conocida por cualquier tipo de usuario como es la web.

3.3.1 OpenRave

El simulador utilizado es OpenRave (Open Robotics Automation Virtual Environment) [26], desarrollado en el Instituto de Robótica de la Universidad Carnegie Mellon, donde se presenta como una arquitectura software multiplataforma de código abierto focalizada en el desarrollo de aplicaciones para el mundo real de robots autónomos.

OpenRave proporciona un entorno para pruebas, desarrollo, e implementación de algoritmos de planificación de movimiento en aplicaciones de robótica del mundo real, e incluye una sólida integración entre simulación, visualización, algoritmos de planificación y control, y leguajes de programación. La atención se centra fundamentalmente en la simulación y el análisis de información cinemática y geométrica relacionada con la planificación de movimientos. La naturaleza independiente de OpenRave le permite ser integrado fácilmente en los sistemas de robótica existentes. También proporciona muchas herramientas de línea de comandos para trabajar con robots y planificadores, y el tiempo de ejecución del núcleo es lo suficientemente pequeño como para ser utilizado dentro de los controladores y los marcos grandes. Una aplicación objetivo importante es la automatización en la robótica industrial.

Presenta una serie de características generales en su diseño:

- Arquitectura basada en múltiples plugins que le confiere mayor funcionalidad y permite que su núcleo sea lo más sencillo posible.

- Un núcleo que puede ser utilizado tanto como entorno físico de simulación 3D, como módulo controlador de las operaciones cinemáticas y dinámicas asociadas a los robots, o como módulo de planificación para tareas de manipulación.
- Un diseño integrado para el control y monitorización de los movimientos ejecutados por los robots en tiempo real.
- Un protocolo de red que proporciona la posibilidad de interactuar con lenguajes de programación interpretados como Matlab, Octave y Python.

El entorno cargado por defecto en el módulo cartesianServer, aquel que laza una instancia de OpenRave, tiene un ambiente del robot asistencial ASIBOT en una cocina de prueba, véase Figura 21. Mientras que para el entorno asistencial de este trabajo se necesita un robot y un paciente, por lo tanto se ha modificado y actualizado un poco el entorno para que se adecúe a las necesidades, véase Figura 22. Los sensores y cámaras del robot también están incorporados en el entorno de simulación 3D para proporcionar un mayor grado de realismo y fidelidad.

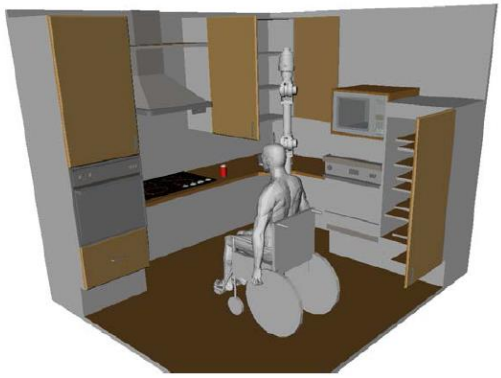

Figura 21. Entorno asistencial realizado anteriormente

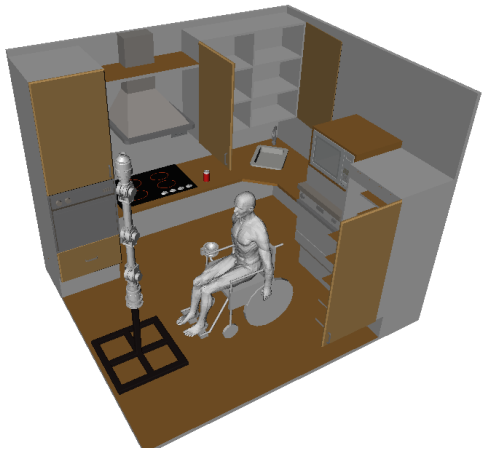

Figura 22. Entorno asistencial actual, con el robot en una base y el paciente enfrente de este para realizar la terapia.

3.3.2 Establecimiento de conexión entre interfaz web y simulador

Una vez sabiendo cual es el simulador, el siguiente paso es conocer su aspecto y modo de empleo, para después poder establecer una comunicación entre la interfaz web y el propio simulador a través de un servidor YARP.

El aspecto de OpenRave se muestra a continuación:

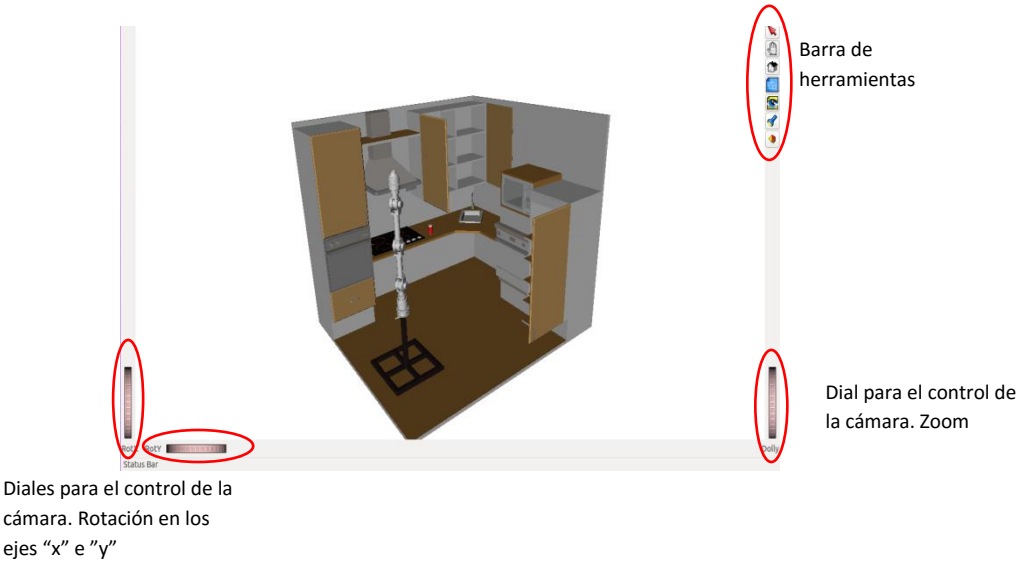

Figura 23. Aspecto de la interfaz de OpenRave

En la barra de herramientas hay varios iconos con diferentes funcionalidades pero los más importantes son los que se exponen a continuación:

Figura 24. Barra de herramientas de OpenRave

Sabiendo el manejo básico del simulador, se procede al establecimiento de la comunicación entre la interfaz web y el simulador. Para ello, primero hay que crear o instanciar un servidor de yarp, que desempeña el papel de enlace de comunicación. Después,

se instancia el simulador, módulo cartesianServer. Y por último se lanza una instancia de la interfaz web, módulo webInterface, para que se active el puerto web y la interfaz pueda ser encontrada por un buscador web.

Teniendo la interfaz web en pantalla, tal y como se ha dicho anteriormente, en la parte inferior izquierda hay un gestor de comunicaciones, que permite enlazar con el robot real o el simulador. En este caso se comunica con el simulador, por tanto se da clic en simulación, véase Figura 25, y automáticamente se enlazan tanto el simulador como la interfaz.

Figura 25. Gestor de comunicaciones de la interfaz web de ASIBOT

Cabe mencionar que de la comunicación no se ha tenido que modificar nada porque está todo correcto y sí se enlazan los elementos deseados.

3.3.3 Movimiento del robot a través de la interfaz

Habiendo comprobado que la comunicación se establece sin problema, lo siguiente es intentar mover el robot desde la interfaz, y comprobar si hay algún problema para abordarlo e intentar solucionarlo.

El movimiento del robot se llevará a cabo desde la pestaña “Joint” de la interfaz web, que tiene el siguiente aspecto.

Figura 26. Pestaña Joint de la interfaz web

Al seleccionar cualquier articulación, a través de “Joypad”, e intentar moverla con las flechas, se produce el siguiente error: el robot se mueve pero al llegar al punto requerido no se para sino que sigue su movimiento, y además de esto, el resto de articulaciones también se mueven consiguiendo que el robot se desensamble por completo. Aparte de desmontarse el robot, las articulaciones se siguen moviendo libremente por el entorno hasta que llega un punto en el que se salen de los límites del mismo, y el simulador se cierra.

Esto no solo sucede con joypad, sino que también ocurre con “Numeric” de la pestaña Joint. Y sucede lo mismo al intentar mover el robot desde la pestaña Cartesian, que son las pestañas diseñadas para el movimiento del robot.

Con los antecedentes que se tiene del error, la solución que se propone es comprobar que los motores físicos en la simulación están desactivados, ya que tienden a ser muy inestables.

En la comprobación del estado de los motores físicos, se encuentra que el pluging RaveBot, que se encarga de crear una instancia de OpenRave e implementar las interfaces de comunicación con yarp de los encoders y el control de posición y velocidad, tiene la física deshabilitada. Por tanto, si no se activan por consola, lo único que queda es revisar que el fichero .XML, que carga el entorno en el simulador, no los activa.

En esta última revisión, se encuentra que el fichero “asibot_kitchen.env.xml” que es cargado en el simulador, activa por defecto los motores físicos, denominados “physicsengine”. Por lo tanto, se procede a eliminar esa activación por defecto, de la siguiente manera. Teniendo en cuenta que se quiere conservar el código fuente desarrollado en un principio, la manera correcta de dicha desactivación, es poner como comentario la parte del código que activa los motores físicos. Quedando de la siguiente manera:

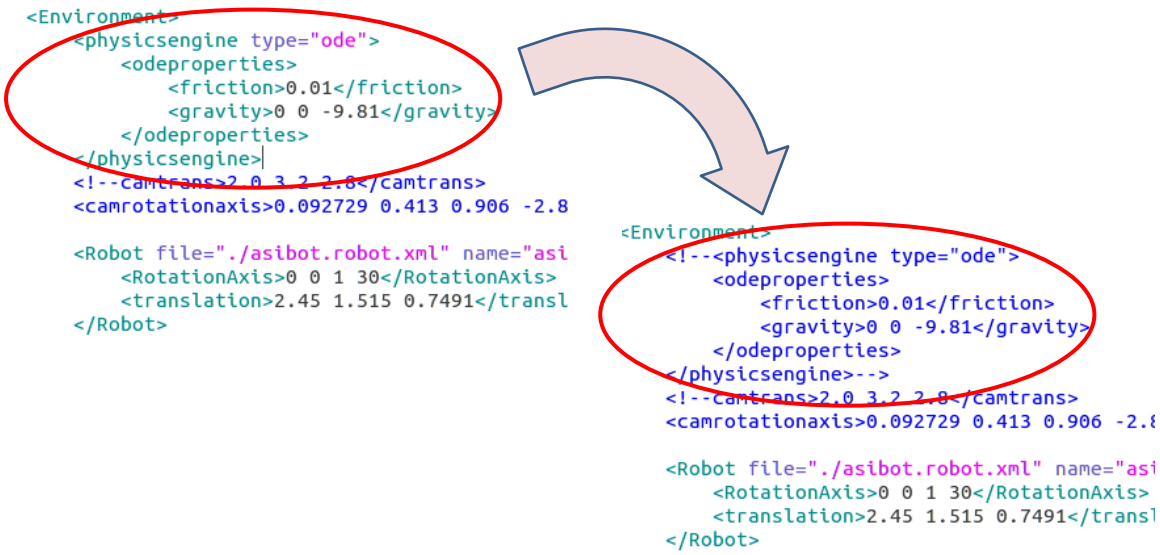

Figura 27. Código fuete actualizado del fichero asibot_kitchen.env.xml

Después de hacer el cambio, se comprueba que la solución aportada es correcta, ya que a la hora de intentar mover el robot desde la interfaz, se consigue satisfactoriamente el objetivo.

3.4 Elaboración de trayectorias para simular ejercicios utilizados en terapia de rehabilitación de miembro superior

El objetivo de este apartado es realizar movimientos del robot desde la interfaz web, con la finalidad de llegar a ciertos puntos claves, guardarlos o capturarlos para posteriormente crear un programa con dichos puntos capturados, consiguiendo de esta manera que el robot realice una trayectoria pasando por cada uno de los puntos anteriormente mencionados.

La finalidad de la elaboración de trayectorias a través de la interfaz web, es conseguir una interacción humano-robot más sencilla, teniendo en cuenta que los usuarios en este caso van a ser los terapeutas, que son unos usuarios no experimentados en el tema de la robótica, más en concreto de la programación robótica.

Cabe destacar que este apartado no se ha conseguido al completo, por tanto, a continuación se irá explicando los objetivos alcanzados y los que no.

3.4.1 Realización de captura de puntos

En este apéndice se expondrá la manera correcta de guardar los puntos clave para la consecución de una trayectoria, y si se ha encontrado un problema al respecto.

Para empezar, los puntos que se van a guardar o capturar, forman parte de la rehabilitación. En concreto, son los puntos donde el brazo robótico se debe parar hasta que el paciente mueva su extremidad superior y llegue al punto donde se encuentra el robot, consiguiendo el objetivo de probar y mejorar la flexo-extensión del brazo del paciente.

Estando en cualquier pestaña de movimiento de la interfaz, por ejemplo en Joint, se mueve el robot hasta el punto deseado. Una vez allí, se da clic en el botón rojo para guardar el punto en el que se encuentra actualmente el robot. Después de haber guardado el punto, con su correspondiente nombre, debe constar dicho punto en la parte derecha, donde pone “Captured Points”, de la pestaña Program. Pero resulta que no se da este caso, por tanto hay que revisar dónde se guardan los puntos, para poder listarlos en la pestaña Program.

Al revisar el fichero WebInterface.cpp, se encuentra que los puntos se guardaban en un directorio inexistente, por tanto se modifica la ruta de guardado para tener accesible el fichero donde constan los puntos que se vayan capturando.

El cambio realizado en el código se puede observar en la siguiente imagen.

```
responder.setResourceFinder(rf);
ConstString userPath = rf.getContextPath() + "/../user/";
ConstString userPath = rf.getContextPath() + "/";
printf("WebInterface using userPath: %s\n",userPath.c_str());
responder.setUserPath(userPath);
ConstString resourcePath = "http://";
```

Figura 28. Cambio de la ruta de guardado de los puntos capturados

Después de realizar el cambio, se puede comprobar que en la pestaña Program ya se listan los puntos capturados, por tanto este objetivo se da por cumplido.

3.4.2 Creación de un programa con los puntos capturados

En este apartado se afronta el tema de la creación de un programa en el que consten los puntos guardados anteriormente y así poder realizar una trayectoria con el brazo robótico.

El programa se desarrolla en lenguaje Python, y además se basa en unas instrucciones de otro lenguaje como es RAPID. Debido a las necesidades de saber lenguajes de programación para poder crear un programa sencillo a través de la interfaz, se ha desarrollado una plantilla (véase figura 29) para facilitar dicha creación, en la que solo se deben añadir los puntos requeridos y ejecutar el programa.

```
from AsibotPy import *
########################################
home=[0,0,1.4,0,0]
########################################
simCart = CartesianClient()
simCart.open('/ravebot')
# use '/canbot' for real
########################################
print 'hello, robot!'
simCart.movl(home)  # defaults to 20 s
simCart.wait()      # wait for movement


########################################
print 'done!'
simCart.close()
```

Figura 29. Plantilla para la creación de un programa en la interfaz web

Teniendo los puntos guardados con anterioridad, lo que se debe hacer ahora es ir a la pestaña Program, en la que ya están listados los puntos capturados, y crear un programa nuevo dándole un nombre.

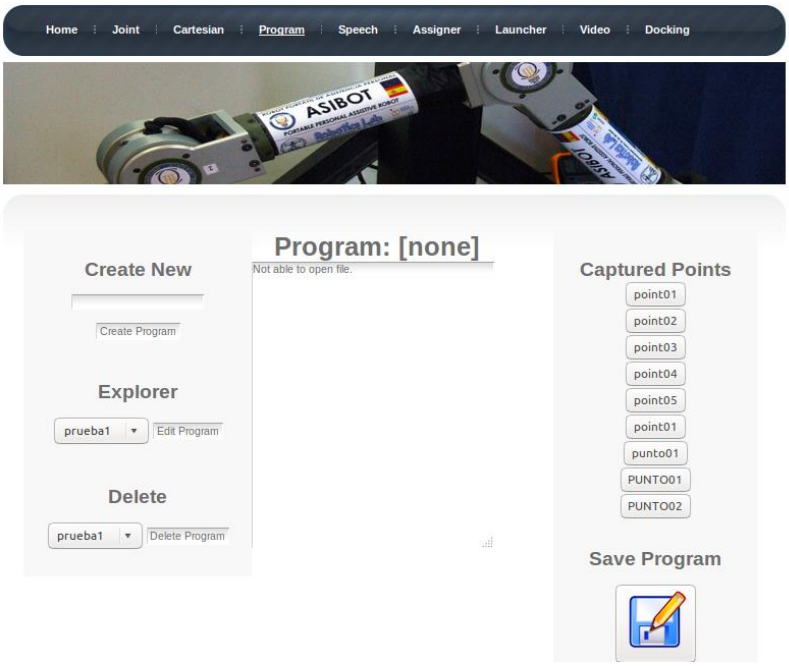

Figura 30. Aspecto de la pestaña Program de la interfaz web

Después de crear el programa, se puede abrir y editar dando clic en la opción de "Edit Program". A continuación, debe salir en la zona central de la pestaña la plantilla mencionada anteriormente. Pero no sucede así, entonces aunque se añadan los puntos al programa, este no se puede ejecutar porque no tiene la estructura esperada ni se incluyen las librerías necesarias para su funcionamiento.

Este problema no se ha podido solucionar todavía, pero se cree que el origen del error es debido a rutas y permisos de directorios y ficheros.

La no consecución de este último objetivo es debido a la falta de tiempo, a la gran diversidad técnica que hay detrás de los recursos utilizados (interfaz web, simulador y comunicaciones entre sí) y al continuo cambio de objetivos del trabajo a lo largo del desarrollo del mismo.

4 RESULTADOS

4.1 Introducción

Este capítulo tiene como objetivo mostrar los distintos resultados y el rendimiento que un usuario final puede obtener al utilizar la interfaz web de ASIBOT haciendo uso de un navegador.

La tarea desarrollada en este trabajo consiste en mover un brazo robótico asistencial en un entorno virtual, para conseguir realizar unas trayectorias que sirvan como ejercicio de rehabilitación para pacientes con problemas motrices de sus extremidades superiores.

Además este trabajo no tiene como objetivo solo poder realizar trayectorias programadas con el robot, si no que para poder realizar esas trayectorias no se necesite de alguien experto en programación, es decir, que la interacción con dicho robot sea lo más natural y sencilla a través de una interfaz web.

A continuación se explicará la apariencia de la interfaz web y el caso de uso de un usuario final.

4.2 Apariencia de la interfaz web de ASIBOT

A continuación se pretende explicar la apariencia de la interfaz web de ASIBOT que tiene nueve pestañas distintas, para que el lector se pueda hacer una idea general de la funcionalidad de la interfaz.

- **Home**. Es la pestaña de inicio de la interfaz web
- **Joint**. Permite mover el brazo robótico en el espacio articular. Pudiendo elegir entre modo incremental (continuo), relativo y absoluto. El modo absoluto tiene como referencia la posición inicial.
- **Cartesian**. Permite movimientos del brazo robótico en el espacio cartesiano. Pudiendo ser también incremental, relativo o absoluto.
- **Program**. Permite crear, leer y editar ficheros de tareas pregrabadas. Esto se consigue leyendo la posición en la que se encuentran los ejes e indexando los datos de los encoders (guarda la posición) en el fichero programa que se esté editando.
- **Speech**. Implementa un reconocimiento de voz, a través del cual se le pueden asignar ciertas tareas al robot.

- **Assigner**. A través de esta pestaña se pueden enviar comandos directamente a las unidades de control de los motores del robot. Pudiendo seleccionar entre programas creados, comandos por voz o iconos de tareas pregrabadas. Véase figura 31.
- **Launcher**. Lista todas las tareas asignadas anteriormente y permite ejecutar o lanzar cualquiera de ellas.
- **Video**. Esta pestaña muestra los flujos de imágenes que publican las cámaras IP que se encuentran situadas dentro del entorno donde trabaja el brazo robótico ASIBOT.
- **Docking**. Permite controlar los solenoides de seguro para el anclaje y des-anclaje del robot de los conectores que utiliza para escalar.

Figura 31. Apariencia de la pestaña Assigner de la interfaz web de ASIBOT

Como se puede observar, la interfaz web pese a implementar controles para una robótica asistencial avanzada, su diseño intuitivo tiene en cuenta que el posible usuario final puede ser un niño, una persona discapacitada o mayor.

4.3 Casos de uso

En este apartado se explicará el modo de uso de la interfaz web, en función de los requerimientos y necesidades de cada usuario.

Como este trabajo está relacionado con una terapia de rehabilitación, se explicará cómo abordar la ejecución de movimientos, la captura de puntos de interés, la creación de un programa y la ejecución del mismo. Sin detenerse en el resto de funcionalidades de la interfaz.

Lo primero que se debe hacer es estudiar la capacidad de flexo-extensión de la extremidad superior del paciente y saber qué tipo de rehabilitación necesita.

Habiendo estudiado al paciente ya se sabe cuál es la máxima extensión del miembro a tratar, por tanto ya se pueden capturar los puntos clave donde el robot debe detenerse.

En este caso se pueden realizar tres tipos de ejercicios de rehabilitación, todas son de extensión del brazo simplemente varían en la trayectoria. Una es flexión frontal, otra de abducción lateral y la última abducción horizontal.

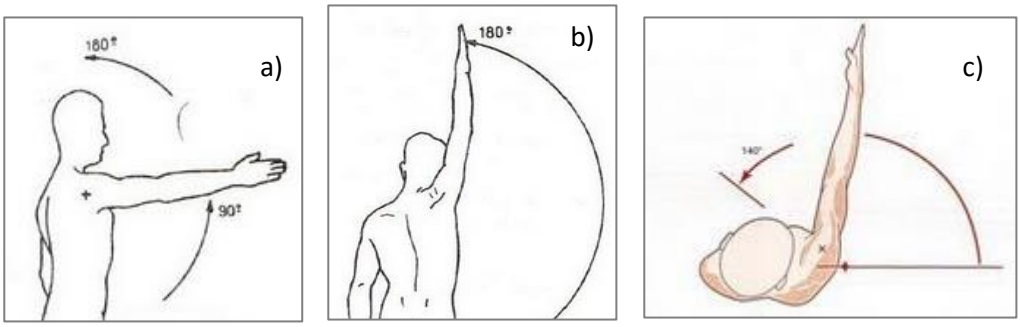

Figura 32. Ejercicios de extensión: a) frontal b) lateral c) horizontal

Sabiendo la trayectoria circular que puede seguir el paciente, se prosigue a la captura de puntos de interés dentro de dicha trayectoria. Para guardar los puntos clave hay que mover el robot hasta dicho punto desde cualquier pestaña de movimiento y capturarlo, dando al botón rojo. Esto se repite hasta conseguir un mínimo de puntos clave para que la rehabilitación sea productiva.

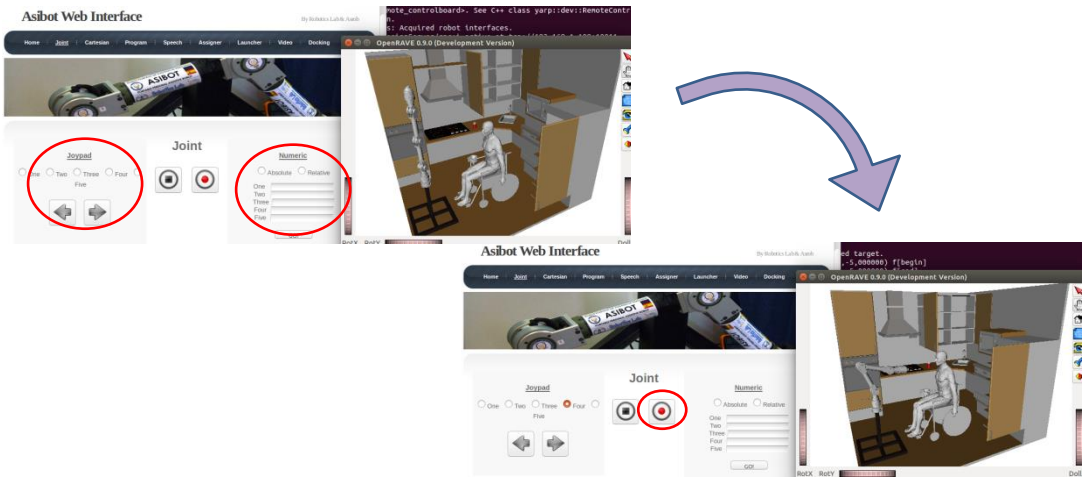

Figura 33. Proceso de captura de puntos a través de la interfaz web de ASIBOT

Se prosigue con la creación de un programa donde consten los puntos guardados anteriormente y unas simples instrucciones para que el robot se mueva de un punto a otro, consiguiendo una trayectoria.

Al ir añadiendo los puntos en el programa, estos se listan uno debajo de otro pero para una mayor claridad se realiza una reordenación de los mismos quedando de la siguiente manera.

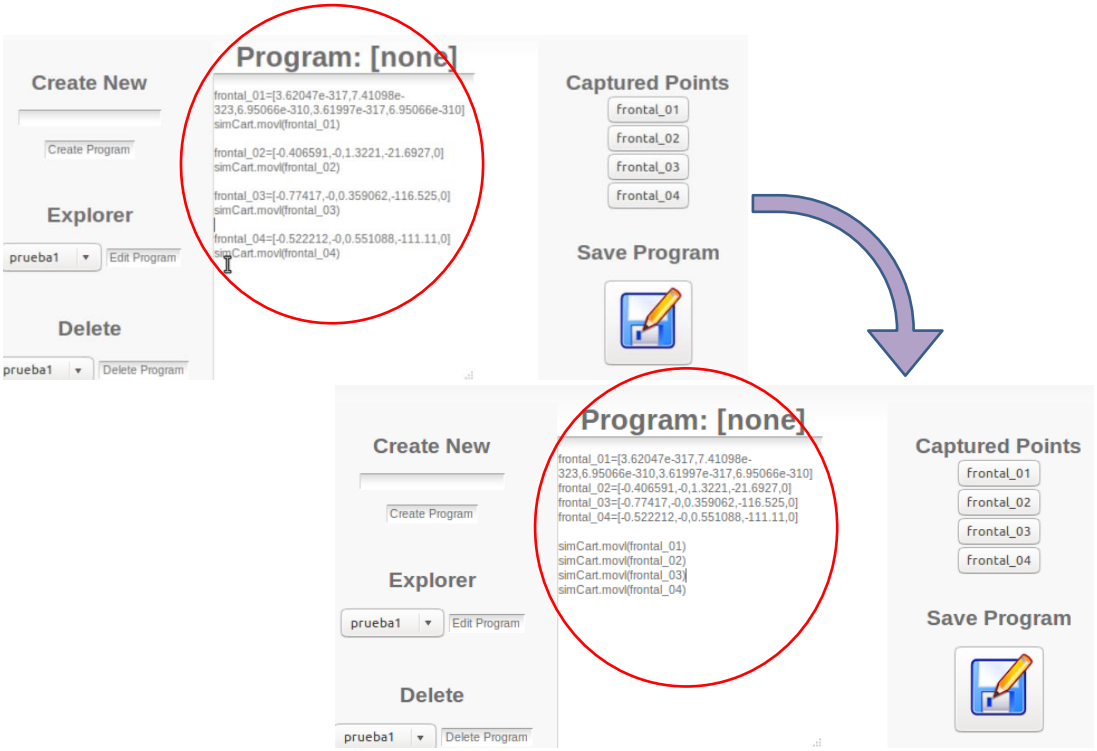

Figura 34. Ordenación de puntos listados en el programa

Cabe destacar que se reordenen los puntos o no, el programa funcionaría de igual manera.

Teniendo el programa creado y guardado el último paso sería ejecutarlo, pero como no se ha llegado a solucionar este problema, se procede a dar una explicación teórica de lo que debería suceder: dentro de la pestaña Assigner en “Select Program” se debe elegir el programa creado y finalmente darle al botón de “Assing to Task” para asignarle dicha tarea, enviando los comandos a la unidad de control de los motores del robot.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

El avance de las tecnologías se incorpora progresivamente a nuestra vida, haciéndonos más fácil muchas de las tareas cotidianas. Actualmente, podemos encontrar casi en cualquier lugar dispositivos que están compuestos por avanzados sistemas electrónicos, como por ejemplo: teléfonos móviles, ordenadores, televisores; que a la vez de implementar controles sofisticados, tienen un manejo natural y sencillo al que todo usuario tiene acceso. Y lo más importante, es que también tienen capacidades de comunicación inalámbrica y acceso a internet. Todos estos avances también se han ido incorporando a la robótica con mayor o menor aceptación por parte del usuario final, dependiendo del ámbito de su aplicación.

La robótica avanza gracias al desarrollo de otros campos de la tecnología como la mecánica, la electrónica y la informática. Gracias al avance de la mecánica, los robots son más ligeros y robustos debido al uso de nuevos materiales como puede ser la fibra de carbono u otros materiales más elásticos y resistentes. Por otra parte, gracias al desarrollo y tratamiento del silicio y materiales semiconductores dentro del campo de la micro-electrónica, el hardware de la robótica es más fiable y rápido.

Todos los avances mencionados anteriormente se han ido integrando en la robótica, consiguiendo una evolución que parte desde el régimen estacionario de las plantas de producción hasta un ámbito más cercano a nosotros como es el doméstico y el de consumo. Algunos ejemplos del amplio alcance de la robótica y la automatización son las persianas automatizadas, aspiradores motorizados, manipuladores robóticos avanzados, e incluso robots de servicios asistenciales.

Sabiendo esto y teniendo en cuenta los resultados del trabajo desarrollado, se puede concluir que la robótica en general puede ser de mucha ayuda para la humanidad, pero si nos centramos en la robótica de servicios, nos podemos dar cuenta que esta tecnología puede integrarse perfectamente en un entorno doméstico para dar mayor bienestar a todos y una mejor integración de los grupos de personas con necesidades especiales. Ya que hay personas que no pueden desarrollar actividades cotidianas como comer o vestirse por sí mismos, teniendo la necesidad de una persona de asistencia.

Por otro lado, si tenemos en cuenta el segundo aspecto en el que se centra este trabajo que es la integración de un brazo robótico en un entorno de rehabilitación. Se puede decir que debido al envejecimiento de la población, el retardo de la jubilación o accidentes fortuitos, sea necesario y fundamental la existencia de programas de terapias robotizadas, prótesis inteligentes y servicios de tele-rehabilitación, entre otros; que asistan a entrenar o

reaprender movimientos, ayuden a incrementar las funciones físicas y cognitivas, y a monitorizar y diagnosticar a personas en su vida diaria.

Y en cuanto al aspecto personal, se pude decir que ha resultado gratamente satisfactorio y enriquecedor el desarrollo de este trabajo, ya que desde el momento de la especialización de la carrera surgieron asignaturas de este ámbito que llamaban mucho la atención y despertaban cierto interés. Durante el desarrollo de este trabajo se ha investigado sobre ciertos temas que eran totalmente desconocidos, por tanto se ha podido comprobar de primera mano la gran importancia que tiene llevar a cabo una investigación previa ante el desarrollo de cualquier trabajo o proyecto.

5.2 Contribución

Con este trabajo se ha intentado contribuir en el avance y desarrollo de la interacción humano-robot a través de interfaces, ya que es un aspecto a tener muy en cuenta en el futuro porque se pueden desarrollar unos robots con funcionalidades increíbles, pero si la interacción con ellos requiere de alguien experto, dicho logro pierde relevancia e impacto en la sociedad.

Por este motivo, se ha retomado un trabajo hecho hace tiempo. Ya que aprovecha muy bien un concepto que está perfectamente aceptado por las personas de hoy en día que es el acceso y la navegación por internet. Y si a esto se le suma la posibilidad de interactuar o controlar un robot a través de una interfaz web, la magnitud del proyecto puede ser ambicioso y muy amplio, ya que el alcance del usuario final se maximiza a toda la sociedad.

5.2 Líneas futuras

A lo largo de la realización de este trabajo han surgido varios inconvenientes, que se han intentado solucionar para conseguir los objetivos marcados al inicio. Pero durante la realización del último objetivo surgió un problema que no se ha podido solventar, por tanto es lo primero que se añade como trabajo futuro.

Posteriormente, se irán añadiendo y detallando tanto mejoras para el trabajo actual como futuras líneas de investigación.

Ejecución de un programa creado a través de la interfaz web de ASIBOT

Para solucionar esta última tarea habría que dedicar tiempo a mirar tanto las rutas de acceso a ficheros creados por la interfaz, como el permiso de los mismos, ya que los avances aplicados en la informática cambian la manera de acceder a ciertos ficheros, y a lo mejor en este caso se usan maneras anticuadas para hacerlo.

Habilitar el reconocimiento de voz de la interfaz web

La versión actual de la interfaz web no tiene habilitada para su uso el reconocimiento de voz, pero dicha funcionalidad puede ser de gran ayuda porque es una forma viable y útil de programar un robot asistencial, teniendo en cuenta la población objetivo de este trabajo. E incluso a parte de una rehabilitación motora, de manipulación o movilidad, puede servir como rehabilitación cognitiva.

Ampliación de la funcionalidad de la terapia

Una vez comprobado el resultado de la terapia de rehabilitación, se pueden tener en cuenta los datos obtenidos de cada terapia para extrapolar y ampliar la funcionalidad de la misma, pudiéndose aplicar también a diagnosis y monitorización. Para conseguir esto se debería dotar al brazo robótico o al entorno de sensores capaces detectar o monitorizar los movimientos del paciente, y por supuesto realizar algoritmos que traten dichos datos capturados.

Ampliación de ámbito

Si los avances de la robótica médica en la cirugía, rehabilitación y prótesis se trasladan a un ámbito asistencial doméstico, se podría obtener una comunicación entre clínica y hogar, consiguiendo lugares de asistencia inteligentes capaces de comunicarse entre sí usando sensores/actuadores ambientales. Cuya finalidad es asistir a personas, ayudarlas en sus tareas cotidianas y mejorar su integración en la sociedad, en conclusión aportar independencia y calidad de vida a personas que no la tienen por sus necesidades especiales.

6 REFERENCIAS

[1] S. Nope, H. Loaiza E. Caicedo. “Programación de un robot bajo el paradigma del aprendizaje por demostración”. Rev. Fac. Ing. Univ. Antioquia Vol. 58. pp. 142-152. Marzo, 2011

[2] A. N. Meltzoff. “Born to Learn: What infants learn from watching us”. The Role of Early Experience in Infant Development. L. A. L. N. A. Fox, and J. G. Warhol (Eds.), Ed. Skillman. Pediatric Institute Publications. New Jersey (USA). 1999. pp. 145-164

[3] B. G. Galef. “Imitation in animals: History, defi nition and interpretation of data from the psychological laboratory”. Social learning: Psychological and Biological Perspectives. T.R. Zentall & B. G. Galef, Jr. (editores.), Ed. Hillsdale. Lawrence Erlbaum. New Jersey (USA). 1988. pp. 3-28.

[4] E. L. Thorndike. “Animal Intelligence”. B. G. Galef (editor) Imitation in animals: History, defi nition and interpretation of data from psychological laboratory. Ed. Macmillan. New York (USA). 1988. pp. 3-28.

[5] S. Nope, H. Loaiza E. Caicedo. “*Modelo Bio-inspirado para el reconocimiento de gestos usando primitivas de movimiento en visión*”. Revista Iberoamericana de Automática e Informática Industrial (RIAI). Vol. 5. 2008. pp. 69-76.

[6] S. Nope, H. Loaiza E. Caicedo. *“Reconstrucción 3D- 2D de Gestos usando Información de Vídeo Monocular Aplicada a un Brazo Robótico*”. Rev. Fac. Ing Univ. Antioquia. Vol. 53. 2010. pp. 145-154.

[7] J. G. Hoyos-Gutiérrez, C. A. Peña-Solórzano, C. L. Garzón-Castro, F. A. Prieto-Ortiz y J. G. Ayala-Garzón, “Hacia el manejo de una herramienta por un robot NAO usando programación por demostración”, Tecno Lógicas, vol. 17, no. 33, pp. 65-76, 2014.

[8] R. Zollner, O. Rogalla, M. Enrenmann, and R. Dillmann, “Mapping Complex Tasks to Robots: Programming by Demonstration in Real-World Environments” Adv. Human-Robot Interact., vol. 14, pp. 119–136, 2005.

[9] V. Kruger, D. Herzog, S. Baby, A. Ude, and D. Kragic, “Learning Actions from Observations,” IEEE Robot. Autom. Mag., vol. 17, no. 2, pp. 30–43, Jun. 2010.

[10] N. Dantam, I. Essa, and M. Stilman, “Linguistic transfer of human assembly tasks to robots,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 237–242.

[11] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, “Learning and generalization of complex tasks from unstructured demonstrations,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 5239–5246.

[12] S. Niekum, S. Chitta, B. Marthi, S. Osentoski, and A. G. Barto, “Incremental Semantically Grounded Learning from Demonstration,” in Robotics: Science and Systems, 2013.

[13] G. Chang and D. Kulic, “Robot task error recovery using Petri nets learned from demonstration,” in 2013 16th International Conference on Advanced Robotics (ICAR), 2013, pp. 1–6.

[14] D. Martinez, G. Alenya, P. Jimenez, C. Torras, J. Rossmann, N. Wantia, E. E. Aksoy, S. Haller, and J. Piater, “Active learning of manipulation sequences,” in 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 5671–5678.

[15] R. Cubek, W. Ertel, and G. Palm, “High-level learning from demonstration with conceptual spaces and subspace clustering,” in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2592–2597.

[16] D. Feil-Seifer and M. J. Mataric, “Defining socially assistive robotics,” in 2005 9th International Conference on Rehabilitation Robotics, 2005, pp. 465 – 468.

[17] T. Fong, I. Nourbakhsh, and K. Dautenhahn. “A survey of socially interactive robots”. Robotics and Autonomous Systems, 42(3-4):143–166, 2003.

[18] C. Winstein, J. Miller, S. Blanton, E. Taub, G. Uswatte, D. Morris, D. Nicols, and S. Wolf. “Methods for a multisite randomized trial to investigate the effect of constraint-induced movement therapy in improving upper extremity function among adults recovering from a cerebrovascular stroke”. Neurorehabilitation and Neural Repair, 2003, 17(3):137–152.

[19] http://www.lavanguardia.com/tecnologia/20160126/301680910899/espana-es-puntera-en-robotica-asistencial-dirigida-a-ancianos-y-enfermos.html

[20] https://es.wikipedia.org/wiki/Rob%C3%B3tica

[21] A. Jardón, M. F. Stoelen, V. Fernández, J. G. Victores, S. Martínez de la Casa, C. Balaguer & F. Bonsignorio. (2011). Aplicación de teoría de la información para el modelado y cuantificación de la interacción persona-robot. *DRT4ALL. IV Congreso Internacional de Diseño, Redes de Investigación y Tecnologías para todos* (págs. 36-38). ISBN:978-84-88934-50-5.

[22] C. Balaguer, A. Gimenez, A. Jardon, A. Sabatini, M. Topping & G. Bolmsjo. (2006). The mats robot: service climbing robot for personal assistance. *Robotics Automation Magazine, IEEE, 13*(1), 51-58.

[23] Bingi, P. and Sharma, M.K., Godla, J.K. 1999. Critical issues affecting an ERP implementation. Information Systems Management−16(3): 7−14.

[24] J.G. Victores, A. Jardon, S. Morante, M.F. Stoelen, S. Martinez, C. Balaguer. (2011). Interacción humano-robot a través de interfaces en la nube. Conference: Robocity2030 9th Workshop, Robots colaborativos e interacción humano-robot

[25] T. Tamei, T. Matsubara, A. Rai, and T. Shibata, "Reinforcement learning of clothing assistance with a dual-arm robot," IEEE-RAS Int. Conf. Humanoid Robot., pp. 733–738, 2011

[26] Diankov, R. (2010, Aug). Automated Construction of Robotics Manipulation Programs. PhD thesis, Robotics Institute: Carnegie Mellon University.